WARWICK
THE UNIVERSITY OF WARWICK

Parameterized Complexity: Permutation Patterns, Graph Arrangements, and Matroid Parameters

by
Lukáš Mach

Thesis

Submitted in partial fulfilment of the requirements
for the degree of *Doctor of Philosophy*

University of Warwick, Department of Computer Science
March 2015

Contents

Acknowledgements

I am grateful to my supervisor, Dan Král', for all the opportunities, support, and guidance provided during my doctoral studies. The feedback provided by my viva examiners, Graham Cormode and Daniel Paulusma, is greatly appreciated.

I want to thank all the researchers who have co-authored a paper with me: Ivan Bliznets, Marek Cygan, František Kardoš, Pavel Klavík, Paweł Komosa, Dan Král', Anita Liebenau, Andrej Mikulík, Michał Pilipczuk, Jean-Sebastien Sereni, and Tomáš Toufar. My bachelor students at Charles University in Prague, Adam Dominec and Viktorie Vášová, were great collaborators as well.

I happily acknowledge the support provided by the following organizations: Computer Science Institute of Charles University in Prague, the Grant Agency of Charles University (GAUK), University of Warwick, the Centre for Discrete Mathematics and its Applications (DIMAP) at Warwick, University of Warsaw, and the Warsaw Center of Mathematics and Computer Science (WCMCS). During my studies, I have received support from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no. 259385 and from the student project GAUK no. 592412.

I want to thank all the great people and friends I have met before and during my studies at Warwick, in particular Rafael Barbosa, Martin Böhm,

Michail Fasoulakis, Matthew Felice-Pace, Tereza Hulcová, Marcin Jurdzinski, Tereza Klimošová, Matjaž Krnc, Lukáš Lánský, Bernard Lidický, Felix Ling, Jirka Matějka, Nick Matsakis, Taísa Martins, Lukáš Novotný, Adéla Peterová, Jakub Polonský, Jakub Sliacan, Vojta Tůma, Danuše Vavřinová, Honza Volec, and Nick Zervoudis.

I would like thank to my family, especially my mother, father, and brother, for all of their love and support. Thanks also go to my dearest Ola.

Declarations

The majority of this thesis is based on results that are either already published or whose publication is currently in preparation. Below, we list these publications including submitted papers (and papers in preparation). The thesis is divided into seven chapters:

- Chapter 1 gives an overview of the obtained results and a commentary on how they relate to the current state of the art in computer science. Chapter 2 provides the reader with background material of the relevant areas and fixes notation used throughout the text. This notation and the statements of definitions, lemmas, hypotheses, and theorems from this chapter are often identical or similar to those used in one of the standard introductory monographs referenced in the text.

- Chapter 3 is based on the result [5]. This collaboration with Ivan Bliznets, Marek Cygan, and Paweł Komosa has been published in *Information Processing Letters*.

- The results presented in Chapter 4 are a joint work [6] with Ivan Bliznets, Marek Cygan, Paweł Komosa, and Michał Pilipczuk. They were accepted for presentation at the SODA 2016 conference.

- Chapter 5 is a joint work with Tomáš Toufar. The corresponding article [64]

is a contribution to the *8th International Symposium on Parameterized and Exact Computation (IPEC 2013)*, where it has been awarded the *IPEC Excellent Student Paper award*. The results of this chapter overlap significantly with the Bachelor thesis of Tomáš Toufar.

- In Chapter 6, we study the algorithmic aspects of the matroid branch-depth parameter. These results were obtained together with František Kardoš, Daniel Král', and Anita Liebenau. The corresponding paper [52] (submitted for a journal publication) introduces this parameter and gives both structural and algorithmic theorems.

- Conclusion of the work with some indication of possible future research directions is given in Chapter 7.

With the exception of the contents of Chapter 5 (mentioned above), no result of this thesis already appeared in another academic thesis. The scientific contribution of the individual authors of the above results is roughly proportional to the number of authors of each result. This thesis has not been previously submitted in any form for any degree at any university.

Abstract

The theory of parameterized complexity is an area of computer science focusing on refined analysis of hard algorithmic problems. In the thesis, we give two complexity lower bounds and define two novel parameters for matroids.

The first lower bound is a kernelization lower bound for the Permutation Pattern Matching problem, which is concerned with finding a permutation pattern inside another input permutation. Our result states that unless a certain (widely believed) complexity hypothesis fails, it is impossible to construct a polynomial time algorithm taking an instance of the Permutation Pattern Matching problem and producing an equivalent instance of size bounded by a polynomial of the length of the pattern. Obtaining such lower bounds has been posed by Stéphane Vialette as an open problem.

We then prove a subexponential lower bound for the computational complexity of the Optimum Linear Arrangement problem. In our theorem, we assume a conjecture about the computational complexity of a variation of the Min Bisection problem.

The two matroid parameters introduced in this work are called amalgam-width and branch-depth. Amalgam-width is a generalization of the branch-width parameter that allows for algorithmic applications even for matroids that are not finitely representable. We prove several results, including a theorem stating that deciding monadic second-order properties is fixed-parameter tractable for general matroids parameterized by amalgam-width. Branch-depth, the other newly introduced matroid parameter, is an analogue of graph tree-depth. We prove several statements relating graph tree-depth and matroid branch-depth. We also present an algorithm that efficiently approximates the value of the parameter on a general oracle-given matroid.

Chapter 1

Introduction

Efficient computation is one of the main focal points of computer science. Since the beginnings of the field, researchers have attempted to either find algorithms minimizing resources exerted for obtaining the correct answer to algorithmic problems or to find negative results – reasons why some problems should not admit algorithms above certain levels of efficiency. The first theoretical notion trying to characterize what is practically computable was the class P of algorithmic problems solvable in polynomial number of steps. From a practical standpoint, this mathematical formalization has several major issues. One particular is that many algorithmic problems widely believed to lie outside of P are actually routinely solved in the real world, e.g. by CSP-solvers. The instances encountered as a result of real applications sometimes seem to posses certain structural properties that help to accelerate the computation. A more detailed look at the (conjectured) land behind the boundaries of P is certainly warranted.

Parameterized complexity, the area to which this thesis mainly contributes, provides a framework for such refined analysis. In this field, the set of all possible inputs of an algorithmic problem (e.g., the set of all graphs) is divided into an

infinite number of layers by equipping the individual instances with *a parameter value.* Different parameterizations of the same set of inputs are sometimes sensible. Ideally, the parameter should, roughly speaking, correspond to the algorithmic difficulty of resolving the instance. One of the main aims of the area is to obtain *fixed-parameter tractable (FPT) algorithms* for parameterized problems. These are algorithms with the running time of $\mathcal{O}\big(f(k) \cdot n^c\big)$, where $f(\cdot)$ is a computable function, n the size of the input of the instance, k the value of the parameter, and c a real constant.

An example of a parameterization is given by the classical notion of *graph tree-width*, employed prominently in the proof of the Robertson-Seymour theorem (see, e.g., [76]). The tree-width of a graph is a value proportional to the "distance" the graph is from being a tree (or a forest). Graphs with tree-width 1 are actually precisely forests. The formal definition is given as Definition 8 in Section 2.3. A large number of difficult (NP-hard) problems on graphs are easy to solve on graphs of bounded tree-width: for any choice of a constant k, there exists a polynomial algorithm solving the problem for graphs of tree-width at most k. Many of the corresponding algorithms are FPT under this parameterization. A classical result of Courcelle [21] generalizes this to all decision problems expressible as formulas in monadic second-order logic.

Theorem 1 (Courcelle, [21])**.** *Let φ be a fixed formula in monadic second-order logic and $k \in \mathbf{N}$. Then, there is an algorithm deciding whether a graph G satisfies φ in linear time for graphs of tree-width at most k.*

Parameterized complexity also provides us with the first formal framework to theoretically analyze preprocessing algorithms through the notion of *kernelization. A preprocessing algorithm* is an algorithm that transforms an instance of

a problem to an equivalent instance of strictly smaller size. Such algorithms are useful when it comes to solving difficult problems in practice: an efficient preprocessing algorithm is applied exhaustively after which another approach is used to solve the reduced instance. Obviously, it is desirable for the size of the reduced instance to be as small as possible. We say that a parameterized problem *has a kernel* if there exists a polynomial time preprocessing algorithm guaranteed to produce an instance of size and parameter value bounded by a function of the parameter of the initial instance. Such algorithm is then called *a kernelization algorithm*. Conveniently, the class of FPT problems and the class of problems with a kernel coincide [27]. A stronger notion is represented by the class of problems with *a polynomial kernel*. For these, there is a kernelization algorithm always producing an instance of size bounded by a polynomial of the parameter value of the initial instance.

The first result of this thesis is a kernel size lower bound for the Permutation Pattern Matching problem. A permutation π contains a permutation σ as a pattern if it contains a subsequence of length $|\sigma|$ whose elements are in the same relative order as in the permutation σ. The Permutation Pattern Matching problem is the corresponding algorithmic problem. The standard parameterization is by $|\sigma|$. Guillemot and Marx [39] recently resolved the issue of whether the problem is in FPT affirmatively, implying that the problem has a kernel. Vialette [82] asked for lower bounds on the size of this kernel. We prove the following:

Theorem 2. *Unless NP $\subseteq$ co-NP/poly, the* Permutation Pattern Matching *problem does not have a polynomial kernel.*

The assumption at the beginning of the theorem is a standard hypothesis

in the field of computational complexity and one of the typical starting points for obtaining kernelization lower bounds. Its failure would imply the collapse of the polynomial hierarchy to the third level. Consequently, there is a high degree of confidence in its validity within the research community. The proof of this theorem is given in Chapter 3.

A lower bound of a different kind is presented in Chapter 4, where we are concerned with the standard (i.e., non-parameterized) Optimum Linear Arrangement problem. We prove a hardness result relative to the computational complexity of the gap version of the Min Bisection problem. (The precise definitions of these algorithmic problems are deferred to Chapter 4.) Specifically, we formulate the following conjecture:

Conjecture 3. *There exist $d_0 \in \mathbf{N}$ and $\alpha, \beta \in (0,1), \alpha < \beta$ such that for each $d \geq d_0$ there is no $2^{o(n)}$ time algorithm for (d, α, β)-Gap Min Bisection.*

The (d, α, β)-Gap Min Bisection problem is defined in Chapter 4. Essentially, it is a gap version of the Min Bisection problem on regular graphs. The motivation for this conjecture is given in Chapter 4. There, we prove the theorem below.

Theorem 4. *Unless Conjecture 3 fails, there is no $2^{o(n+m)}$ time algorithm for* Optimum Linear Arrangement*, where n is the number of vertices of the input graph and m the number of its edges.*

This is a similar kind of computational complexity bound to those based on *the Exponential Time Hypothesis (ETH)*, which stipulates that there is no algorithm for the q-SAT problem running in $2^{o(n)}$ time, where n is the number of variables of the input formula. In fact, as a part of an ongoing research project

reported in [6] we prove complexity lower bounds for several *graph completion problems* first under the ETH and then apply Conjecture 3 and Theorem 4 to substantially strengthen them.

Let us briefly comment on some results from [6] that are not part of this thesis. Hopefully, they provide additional level of justification for our conjecture and illuminate the relevance of Theorem 4. Graph completion problems are algorithmic problems where the question to be decided is whether a certain number of edges can be added to the input graph in such a way that the resulting graph belongs to some class, e.g., the class of chordal graphs. We consider the natural parameterization of these problems by the number of edges to be added and give the following bounds. (We refer the reader to Section 2.4 for the definitions of the algorithmic problems in question.)

Theorem 5 (Bliznets et al., [6])**.** *Unless the ETH fails, there exists $c > 0$ such that none of the following problems has a $2^{\mathcal{O}(k^{\frac{1}{4}}/\log^c k)} \cdot n^{\mathcal{O}(1)}$ algorithm, where k is the value of the parameter and n is the input size:* Chordal Completion*,* Trivially Perfect Completion*,* Proper Interval Completion*, and* Interval Completion*.*

The parameterized problems from the above theorem have been studied extensively in the literature [34]. Theorem 4 allows us to obtain lower bounds that essentially match the fastest presently known parameterized algorithms.

Theorem 6 (Bliznets et al., [6])**.** *Unless Conjecture 3 fails, for all $\varepsilon > 0$ there is no $2^{\mathcal{O}(\sqrt{k^{1-\varepsilon}})} \cdot n^{\mathcal{O}(1)}$ algorithm for any of the following problems:* Chordal Completion*,* Trivially Perfect Completion*,* Proper Interval Completion*, and* Interval Completion*.*

For example, the currently fastest known parameterized algorithm for Chordal Completion runs in $2^{\mathcal{O}(\sqrt{k}\log k)} + n^{\mathcal{O}(1)}$ steps [34].

In Chapter 5, we introduce the notion of amalgam-width. This is an attempt to generalize the notion of width parameters to matroids. Several such parameters have already been introduced in the literature, including a direct generalization of tree-width [44] and clique-width [22]. However, our aim is to obtain one that allows the algorithmic applications even for non-representable matroids. Furthermore, amalgam-width has the nice property that it corresponds to a natural, standard gluing operation called *amalgamation*. We prove the following theorem:

Theorem 7. *For each $k \in \mathbf{N}$ and each formula φ in monadic second-order logic there is an algorithm deciding whether a matroid M satisfies φ in linear time for matroids of amalgam-width bounded from above by k (assuming the corresponding amalgam decomposition $\mathcal{T}$ of the matroid is given explicitly as a part of the input).*

Amalgam-width generalizes a parameter based on 2-sums introduced in [81] and branch-width for matroids representable over finite fields. The later generalization is in the sense that amalgam-width can be bounded by a function of branch-width. The proof of this claim (Proposition 46) is constructive and leads to an efficient algorithm constructing such amalgam decompositions.

The next chapter introduces another matroid parameter, the matroid branch-depth. This time, the main motivation for our algorithmic results is structural. The parameter is a matroid analogue of graph tree-depth, an established graph parameter used among others by Nešetřil and Ossona de Mendez in their research on combinatorial limits [66]. There, the notion of tree-depth is used to constrain the elements of infinite convergent sequences of graphs so that the existence of

a combinatorial limit called modeling can be guaranteed. In [52] we extend this theory to matroids by introducing matroid modelings, the branch-depth parameter, and the notion of first-order convergent sequences of matroids. Apart from a number of negative results, we also prove a theorem analogous to the abovementioned result on graphs. Specifically, we show that every first-order convergent sequence of matroids with bounded branch-depth representable over a finite field has a matroid modeling. The similarity between this statement and the aforementioned result of [66] indicates that at least from some perspective the branch-depth parameter is an analogous notion to tree-depth.

In Chapter 6 of this thesis, we focus exclusively on the algorithmic aspects of the branch-depth parameter and on its relationship with tree-depth. We provide an efficient algorithm approximating the parameter value for any oracle given matroid. The close relation with tree-depth is substantiated by the following properties:

- The branch-depth of a graphic matroid $M(G)$ is at most the tree-depth of G. Furthermore, it can be bounded from below by a function of the tree-depth if G is 2-connected.
- Both branch-depth and tree-depth are minor monotone parameters.
- The branch-depth of a matroid is at most the square of the length of its largest circuit (recall that the tree-depth of a graph G is at most the length of its longest path).
- The branch-depth of a matroid is at least the logarithm of the length of its largest circuit (note that the tree-depth of a graph G is at least the logarithm of the length of its longest path).

The branch-depth parameter is defined through a particular kind of decomposition (Definition 51 in Section 6.1).

Chapter 2

Preliminaries

In this chapter, we review the notation adopted by this thesis and reference results from literature utilized in the proofs of the subsequent chapters. The contents are not intended as an introductory text to the field. Rather, we hope to provide a sufficient amount of context for our results while maintaining brevity. Many of the elementary notions are mentioned simply to fix the notation. We refer the reader to the monographs [25] and [71] for a comprehensive introduction to graph theory and the theory of matroids, respectively. Parameterized algorithms are treated in the recent book [23].

2.1 Basic notation

The set $\{i, i+1, \ldots, j-1, j\}$ is denoted by $[i, j]$. We let $[n] := [1, n]$. The set of all subsets of the set X is denoted by $\mathcal{S}(X)$ and $\binom{X}{r}$ is the set of all subsets of X of size $r \in \mathbf{N}$. For a function f and a set X we define $f(X)$ to be the set $\{f(x) : x \in X\}$. For a matrix M, the element in the i-th row and the j-th column is denoted by $M_{i,j}$. A submatrix of a matrix M is a matrix obtained by deleting

some rows and/or columns of M. For a non-empty set Σ, we denote by Σ^* the set of all sequences of finite length composed of elements from Σ. For example $\{0,1\}^*$ is the set of all finite binary strings. A formula in *conjunctive normal form (CNF)* is a formula $C_1 \wedge C_2 \wedge \ldots \wedge C_k$, where C_i are *clauses* of the form $C_i = \ell_1^i \vee \ell_2^i \vee \ldots \vee \ell_{k_i}^i$, where ℓ_j^i is a literal (either a variable or a negated variable). This formula is *a q-CNF formula* if all its clauses have precisely q literals.

2.2 Permutations

A permutation π is a bijection from $[n]$ to $[n]$. The value $\pi(i)$ is called *the entry of π at position i* (or *index i*). We use $|\pi|$ to denote the size of the domain of π. Two common representations of a permutation π are used in the thesis: the vector $(\pi(1), \pi(2), \ldots, \pi(n))$ and the corresponding permutation matrix. The latter is a $|\pi| \times |\pi|$ binary matrix with 1-entries precisely on coordinates $(\pi(i), i)$.

2.3 Graphs

A graph is a pair (V, E), where V is the set of *vertices* and $E \subseteq \binom{V}{2}$ is the set of *edges*. Therefore, unless otherwise specified, graphs in this thesis are undirected, loopless, and without edge multiplicities. In line with common practice, an edge $e = \{u, v\}$ is denoted by (u, v) although the pair is not ordered. We say that an edge e is *incident with the vertex v* if $v \in e$. The set of vertices of G is denoted $V(G)$ while $E(G)$ is its edge-set. *A hypergraph* is a pair (V, E), where $E \subseteq \mathcal{S}(V)$. *An r-uniform hypergraph* is a hypergraph where $E \subseteq \binom{V}{r}$. Therefore, the notions of a graph and a 2-uniform hypergraph coincide. A graph H is *a subgraph of G* if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. The subgraph H is said to be *induced* if

$E(H) = E(G) \cap \binom{V(H)}{2}$. We use $G[X]$ to denote an induced subgraph of G with vertex set X. *A complete graph K_n on n vertices* is the graph $(V, \binom{V}{2})$, where $V = [n]$. A graph G contains *a clique of size l* if an isomorphic copy of K_l is a subgraph of G.

For a vertex v, $\deg(v)$ is its degree: the number of edges incident with v. We say that a graph is *d-regular* if all its vertices have degree d. We use $\Delta(G) := \max\{\deg(v) : v \in V(G)\}$ to denote the maximum degree of G. The set $N(v) := \{w : (v, w) \in E(G)\}$ is the neighbourhood of v. This notation is extended to subsets of vertices X, i.e. $N(X) = \bigcup_{v \in X} N(v) \setminus X$.

If $X, Y \subseteq V$ are disjoint, then $E(X, Y)$ is the set of edges between X and Y. We use $G[X, Y]$ to denote *the induced bipartite subgraph of G with parts X and Y*. That is, $G[X, Y]$ is the graph with vertex set $X \cup Y$ that contains precisely the edges $E(X, Y)$. For $X \subseteq V$, we denote by $\delta(X)$ the set of edges with exactly one endpoint in X. *A bipartition of a graph G* is a pair (A, B), where $A, B \subseteq V(G), A \cap B = \emptyset$, and $A \cup B = V(G)$. *A balanced bipartition of a graph G* is a bipartition (A, B) such that $\big||A| - |B|\big| \leq 1$. We define *a cut* as the set of edges $E(A, B)$, for a bipartition (A, B) of V. *A balanced cut* is the set $E(A, B)$ where (A, B) is balanced bipartition. The number $|E(A, B)|$ is called *the size of the cut*. *A complement of G* is the graph with the vertices V and edges $\binom{V}{2} \setminus E$. We denote it by $\overline{G}$. We add subscripts to the above notation, e.g., $\deg_G(v), N_G(X), \delta_G(X)$, or $E_G(U, V)$, when it is not clear from context which graph we refer to.

A dominating set of a graph G is a subset X of the vertex set $V := V(G)$ such that $X \cup N(X) = V$. *Chordal graphs* are graphs where every cycle (of length at least four) has *a chord* – an edge connecting two vertices of the cycle which is

not a part of the edge-set of the cycle. *An interval graph* is a graph G such that $V(G)$ can be placed in correspondence to a set of intervals on the real line and two vertices are connected precisely when they have a non-empty intersection. (We also use the term *interval graph* for graphs isomorphic to such graphs.) *A proper interval graph* is an interval graph where no pair of vertices corresponds to a pair of intervals such that one properly contains the other. *A trivially perfect graph* is an interval graph where each pair of vertices corresponds to a pair of intervals that are either disjoint or one contains the other.

A k-tree is either a clique of size $k+1$ or a graph that can be obtained from a smaller k-tree by adding a new vertex and connecting it to k vertices that already form a clique.

Definition 8. Tree-width of a graph G *is the least number* $k \in \mathbf{N}$ *such that* G *is a subgraph of a* k*-tree.*

2.4 Algorithms

Formally, *an algorithmic problem* is a mapping $f : \{0,1\}^* \to \{0,1\}^*$ prescribing each binary *input* the appropriate *output*. We are interested in the construction of Turing machines encoding the function f while simultaneously satisfying certain requirements, typically on running time. Such a Turing machine (and the corresponding *algorithm*) is said to *solve* the particular problem f. Complementarily, we sometimes try to find reasons why a Turing machine with certain properties should not exist. For those arguments, additional hypotheses are typically required.

A decision problem is a special kind of algorithmic problem where the set of values of the mapping is simply $\{0,1\}$. The instances mapping to 1 are sometimes termed *the "YES" instances*, while the rest are *"NO" instances*. A decision problem can be identified with the subset of inputs $\{0,1\}^*$ where the mapping f from the above definition attains the value of 1. Such a set is often called *a language*. We say that the Turing machine solving the decision problem in question *recognizes* this set. Since this thesis is not concerned with the low-level implementation details of Turing machines, we abuse the notation and identify the finite binary sequences with the discrete structures (graphs, permutations, formulas, subsets of vertices, etc.) they are representing. The particular choice of representation is almost never an issue as long as it is reasonably efficient.

We now give an overview of the algorithmic problems that appear in this thesis. SAT, Clique, and Dominating Set are well known NP-hard problems:

SAT

Input: A formula φ in the conjunctive normal form.

Question: Is there an assignment of values to the free variables of φ satisfying the formula?

Clique

Input: A graph $G = (V, E)$, an integer k.

Question: Does G contain a clique of size k?

DOMINATING SET

Input: A graph $G = (V, E)$, an integer k.

Question: Does G contain a dominating set of size k?

There are many variations on the SAT problem in the literature. Of particular importance is the q-SAT problem, where $q \in \mathbf{N}$.

q-SAT

Input: A formula φ in the conjunctive normal form where each clause contains q distinct literals.

Question: Is there an assignment of values to the free variables of φ satisfying the formula?

Graph completion problems are a particular type of decision problems where we ask whether a number of edges can be added to an input graph so that the resulting graph belongs to a certain class of graphs, e.g., chordal graphs. The following completion problems are NP-hard. Their parameterized variants (where k is the parameter) are in FPT.

CHORDAL COMPLETION

Input: A graph G, an integer k.

Question: Can k edges be added to G so that G is a chordal graph?

TRIVIALLY PERFECT COMPLETION

Input: A graph G, an integer k.

Question: Can k edges be added to G so that G is a trivially perfect graph?

PROPER INTERVAL COMPLETION

Input: A graph G, an integer k.

Question: Can k edges be added to G so that G is a proper interval graph?

INTERVAL COMPLETION

Input: A graph G, an integer k.

Question: Can k edges be added to G so that G is an interval graph?

2.5 Complexity classes

A complexity class is a (typically infinite) set of decision problems. The two most fundamental complexity classes are P and NP. The former is the set of all decision problems solvable by a Turing machine in polynomial time. The latter is a set of decision problems that can be decided in polynomial time when an appropriate certificate (i.e., an appropriate input dependent string of bits with polynomial length) is given to the Turing machine in addition to the input. If C is a complexity class, then co-C is the class

$$\Big\{\{0,1\}^* \setminus X : X \in C\Big\}.$$

For example, co-NP is the class of all problems, where the "NO" instances can be identified in polynomial time when an appropriate certificate is provided.

Deciding a problem in NP is equivalent to determining if there exists a polynomially long string of bits satisfying a certain property from P. When resolving a problem in co-NP, we are effectively checking whether a certain property from

P holds for all such strings. This leads to a method that takes a class of problems and generates a definition of a new complexity class. Iterating this gives us *the polynomial hierarchy* of complexity classes. Structural complexity is not the focus of this thesis and we reference the polynomial hierarchy in only a limited number of places to provide basic information about how certain conjectures fit within the rest of the theory. We therefore omit the precise definition of the hierarchy and only recall two key properties. Firstly, the base level of the hierarchy consists of the class P with the next level being formed by NP and co-NP. Secondly, any class from the hierarchy is a superset of each of the classes on the levels beneath it. These inclusions are widely believed to be strict. The famous $\mathsf{P} \neq \mathsf{NP}$ conjecture corresponds to a strict inclusion between the first two levels of the polynomial hierarchy.

We can also define a so called *non-uniform class C/poly* for each complexity class C. A decision problem L is in C/poly if there exists a set $A \in C$ and *an advice function* $g : \mathbf{N} \to \{0,1\}^*$ with the properties:

1. there exists $k \in \mathbf{N}$ such that $|g(n)| \leq n^k$ for all $n \in N$,
2. $x \in L \Leftrightarrow (x, g(|x|)) \in A$.

2.6 Common hypotheses in complexity theory

The following two hypotheses are routinely assumed to hold in the area of computational complexity:

Hypothesis 9. *$\mathsf{P} \neq \mathsf{NP}$.*

Hypothesis 10. *$\mathsf{NP} \not\subseteq$ co-NP/poly.*

Of course, Hypothesis 9 is the more likely answer to the central question of computer science. Its failure would imply the collapse of the entire polynomial hierarchy, something generally considered quite unlikely. Hypothesis 10 is slightly stronger. Its failure would imply the collapse of the polynomial hierarchy to the third level [20]. Still, even this stronger hypothesis is considered to be very safe and is often used by researchers as a starting point from which complexity lower bounds are derived. Yet another standard complexity hypothesis is discussed in the following section.

2.7 Exponential Time Hypothesis

The trivial approach of solving the 3-SAT problem where the input formula φ has n variables and m clauses uses $\mathcal{O}(m2^n)$ steps. Although some improvements on this are possible, the current state of research strongly suggests that this exponential time complexity is unavoidable. This leads us to the following hypothesis, which is called *the Exponential Time Hypothesis (ETH)* [49].

Exponential Time Hypothesis. *The infimum of the set of constants c for which there exists an algorithm solving 3-*SAT *in time $\mathcal{O}^*(2^{cn})$ is strictly larger than zero.*

The ETH asserts that 3-SAT cannot be solved in $2^{o(n)}$ time. (There is a stronger variant of the ETH, called *the Strong Exponential Time Hypothesis* [50]. However, the confidence in its validity is significantly weaker than the confidence in the ETH.)

As already mentioned in the previous chapter, the ETH can be employed to derive (subexponential) lower bounds on the time complexity of various algo-

rithmic problems. Suppose there is an efficient reduction from 3-SAT to some decision problem Q. An algorithm solving Q "too quickly" would then violate the ETH. The precise lower bound on the runtime of any algorithm solving Q obtained in this way depends on the properties of the reduction. Specifically, we are concerned with the size of the instances of Q generated from an instance of 3-SAT with n variables. Reductions that are guaranteed to generate smaller instances result in stronger bounds on the complexity of Q.

Two issues come into the picture. Firstly, there might be an inherent blow-up in the size of the instances when reducing to Q. For example, a 3-SAT instance with n variables and m clauses might require instances of Q of size at least n^2. Secondly, we do not have exact knowledge of the number of clauses of the instance of 3-SAT and thus can only assume $m \leq n^3$. Since polynomial reductions from 3-SAT often explicitly encode the individual clauses of the formula, this could again result in a weaker bound. Fortunately, the second problem can be dealt with through *the Sparsification Lemma*:

Theorem 11 (Sparsification Lemma, [50])**.** *For any $\varepsilon > 0$ and $q > 0$, there exists a constant $C = C(\varepsilon, q)$ such that any q-CNF formula φ with n variables can be expressed as $\bigwedge_{i=1}^{t} \psi_i$, where $t \leq 2^{\varepsilon n}$ and each ψ_i is a q-CNF formula with the same variable set as φ and number of clauses bounded by Cn. Moreover, this formula can be constructed in $\mathcal{O}^*(2^{\varepsilon n})$ steps.*

When combined, the Exponential Time Hypothesis and Sparsification Lemma imply there is no $2^{o(n+m)}$ algorithm for 3-SAT. A reduction from 3-SAT to some decision problem Q that results in instances of size bounded from above by $f(n+m)$ therefore implies there are no algorithms recognizing Q in $2^{o(f^{-1}(n+m))}$ steps (assuming the ETH).

2.8 Expanders

Expanders are sparse graphs that behave in a random-like manner. Their key property is that if we look at any subset X of the vertex set of such a graph, we are guaranteed to see a lot of edges going between X and the remainder of the vertex set. In this work, expanders are used as a black box for a construction in Chapter 4.

The Cheeger number $h(G)$ of a graph G is the quantity

$$h(G) := \min\left\{\frac{|\delta(X)|}{|X|} : X \subseteq V(G), |X| \leq \frac{|V(G)|}{2}\right\}.$$

We say that a graph G is *a (d, e)-expander* if it is d-regular and has $h(G) \geq e$. The following theorem provides us with an explicit construction of expander graphs [45].

Theorem 12. *For every prime p and every $k \in \mathbf{N}$ we can construct in polynomial time $(d, \frac{d-2\sqrt{d-1}}{2})$-expanders, where $d = p^k + 1$.*

We use $G_{n,d}$ to denote a d-regular expander graph on n vertices such that

$$h(G) \geq \frac{d}{3}.$$

The above theorem provides us with a polynomial deterministic algorithm constructing $G_{n,d}$ for some values of d.

2.9 Parameterized complexity and kernelization

A parameterized problem is a set $Q \subseteq \Sigma^* \times \mathbf{N}$, where Σ is a fixed alphabet. The value k of the instance $(x, k) \in Q$ is its *parameter*. A problem Q is in FPT if there is an algorithm deciding $(x, k) \in Q$ in time $f(k)|x|^{\mathcal{O}(1)}$, where f is a computable function. Similarly to the situation in "standard" complexity theory, a hierarchy of classes of parameterized problems can be introduced along with a suitable notion of fixed-parameter tractable reductions. One of such standard hierarchies is the W-hierarchy, introduced by Downey and Fellows [27]:

$$\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \ldots \subseteq \mathsf{W}[P].$$

Similarly to the situation with the polynomial hierarchy discussed in Section 2.5, we are not focusing on the structural aspects of parameterized complexity in this work and the W-hierarchy is referenced only in a handful of places to provide additional context. The reader is referred to [33] for a detailed structural treatment of the theory of parameterized complexity, the precise definitions of the above complexity classes, and the notion of fixed-parameter tractable reductions. For our work, the following properties of the hierarchy are relevant. The above inclusions are conjectured to be strict. In fact, the ETH discussed in Section 2.7 implies $\mathsf{FPT} \neq \mathsf{W}[1]$. The Clique problem (parameterized by the size of the clique) is complete for the class $\mathsf{W}[1]$, while the Dominating Set problem (parameterized by the size of the dominating set) is complete for $\mathsf{W}[2]$.

Algorithms solving a decision problem are required to either output YES or NO. This can be relaxed by not requiring the algorithm to output a definite answer but rather to generate an equivalent instance of the same problem, ideally

one that is as small as possible. This leads us to the notion of *kernelization algorithms*, which we have already discussed in the previous chapter. Below, we provide a rigorous definition.

Definition 13. A kernelization algorithm for a parameterized problem Q *is an algorithm that given an instance* $(x, k) \in \Sigma^* \times \mathbf{N}$ *produces in* $p(|x| + k)$ *steps an instance* (x', k') *such that*

1. $(x, k) \in Q \Leftrightarrow (x', k') \in Q$ *and*

2. $|x'|, k' \leq f(k)$,

where $p(\cdot)$ *is a polynomial and* $f(\cdot)$ *a computable function.*

If there is a kernelization algorithm for Q, we say that Q *has a kernel*. If the function $f(\cdot)$ in the above definition can be bounded by a polynomial, we say that Q *has a polynomial kernel*.

There is a simple relationship between the class FPT and problems with a kernel given by the theorem below.

Theorem 14 (Theorem 1.39 in [33])**.** *For every parameterized problem* Q*, the following are equivalent:*

1. Q *is in* FPT*.*

2. Q *is decidable and has a kernel.*

A result of Bodlaender et al. [8], which builds on [7, 35], is often used to derive kernelization lower bounds under the widely believed complexity assumption $\mathsf{NP} \not\subseteq \text{co-}\mathsf{NP}/\text{poly}$ from Hypothesis 10. As stated above, the current state of research strongly suggests this hypothesis holds.

We start a brief exposition of this standard machinery with two technical definitions.

Definition 15 (Bodlaender et al., [8]). *An equivalence relation R on Σ^* is called* a polynomial equivalence relation *if the following two conditions hold:*

1. *There is an algorithm that given two strings $x, y \in \Sigma^*$ decides whether x and y belong to the same equivalence class in $(|x| + |y|)^{\mathcal{O}(1)}$ time.*

2. *For any finite set $S \in \Sigma^*$ the equivalence relation R partitions the elements of S into at most $(\max_{x \in S} |x|)^{\mathcal{O}(1)}$ equivalence classes.*

An example of such a relation is the grouping of instances of the same size.

Definition 16 (Bodlaender et al., [8]). *Let $L \subseteq \Sigma^*$ be a set and let $Q \subseteq \Sigma^* \times \mathbf{N}$ be a parameterized problem. We say that L* cross-composes into *Q if there is a polynomial equivalence relation R and an algorithm which, given t strings $x_1, x_2, \ldots, x_t$ belonging to the same equivalence class of R, computes an instance $(x^*, k^*) \in \Sigma^* \times \mathbf{N}$ in time polynomial in $\sum_{i=1}^{t} |x_i|$ such that:*

1. *$(x^*, k^*) \in Q \Leftrightarrow x_i \in L$ for some $1 \leq i \leq t$,*

2. *k^* is bounded by a polynomial in $\max_{i=1}^{t} |x_i| + \log t$.*

As the following theorem states, widely believed complexity assumptions imply that it is unlikely for an NP-hard problem to cross-compose into a parameterized problem with a polynomial kernel. The reason behind this is that it would then allow us to find a satisfiable instance of SAT among a very large set of SAT-instances by dividing it in two halves, cross-composing the first half of the instances, kernelizing the resulting instance, solving the resulting instance, and then recursing into the first or second half of SAT instances based on the result.

Of course, the running time of this routine is not polynomial but the existence of such a "detection" algorithm is still something that is generally considered very unlikely for SAT.

Theorem 17 (Bodlaender et al., [8]). *Let $L \subseteq \Sigma^*$ be an NP-hard language. If L cross-composes into the parameterized problem Q and Q has a polynomial kernel then $\mathsf{NP} \subseteq$ co-NP/poly.*

2.10 Matroids

We now present the basic concepts and definitions of the theory of matroids. *A matroid* M is a tuple $(E, \mathcal{I})$ such that E is finite and $\mathcal{I} \subseteq \mathcal{S}(E)$ is the set of *independent sets of* M. The set $\mathcal{I}$ is not a general set system – rather, it is required to satisfy the following matroid axioms. Firstly, $\mathcal{I}$ is not empty. Secondly, it must contain as elements all subsets of any independent set (including the empty set). Finally, the set $\mathcal{I}$ must satisfy *the exchange axiom*:

$$\forall F, F' \in \mathcal{I} \text{ satisfying } |F| < |F'| \text{ there is } x \in F' : F \cup \{x\} \in \mathcal{I}.$$

The set E is termed *the ground set*. For a general matroid M, we denote the ground set by $E(M)$ and call its elements *the elements of* M. If a set is not independent, we call it *dependent*. Any minimal dependent set is called *a circuit*. The set of all circuits of the matroid, denoted by $\mathcal{C}(M)$, uniquely determines the matroid. Any maximal independent set is *a basis of* M. We note that while we only work with finite matroids in this thesis, there are generalizations of the theory to infinite matroids [13].

The rank $\mathrm{r}(F)$ of a set $F \subseteq E(M)$ is the size of the largest $I \subseteq F$ such

that $I \in \mathcal{I}$. *The closure operator* $\mathrm{cl}(F)$ acting on subsets of $E(M)$ is defined as $\mathrm{cl}(F) := \{x : \mathrm{r}(F \cup \{x\}) = \mathrm{r}(F)\}$. It can be shown that $\mathrm{r}(\mathrm{cl}(F)) = \mathrm{r}(F)$. *A loop* of M is an element e such that $\{e\}$ is a circuit (alternatively, $\mathrm{r}(\{e\}) = 0$) and *a bridge* is an element such that $\mathrm{r}(M \setminus \{e\}) = \mathrm{r}(M) - 1$. A set F satisfying $\mathrm{cl}(F) = F$ is called *a flat*.

By $M \setminus F$ we denote the matroid resulting from deleting the elements of $F \subseteq E(M)$: the elements of $M \setminus F$ are $E(M) \setminus F$, with $F' \subseteq E(M \setminus F)$ being independent in $M \setminus F$ if and only if it is independent in M. The restriction $M|F$ of M to F is the matroid $M \setminus \bar{F}$, where $\bar{F}$ denotes the complement of F in $E(M)$. Similarly to graphs, we also define *element contraction*: the matroid M/F is a matroid with ground set $E(M) \setminus F$ where a set $F' \subseteq E(M) \setminus F$ is independent in M/F if and only if $\mathrm{r}(F \cup F') = \mathrm{r}(F) + \mathrm{r}(F')$. Matroid is *a minor of a matroid* M if it can be obtained from M by a sequence of element deletions and contractions.

We refer to any bipartition of $E(M)$ into A and B as *a separation* (A, B). The *size of the separation* (A, B) is equal to $\mathrm{r}(A) + \mathrm{r}(B) - \mathrm{r}(M) + 1$. A separation of size at most k is *a k-separation*. A matroid M is *connected* if the only two subsets $F \subseteq E(M)$ satisfying $\mathrm{r}(F) + \mathrm{r}(\bar{F}) = \mathrm{r}(M)$ are the empty set and $E(M)$. A *component of* M is an inclusion-wise maximal set $F \subseteq E(M)$ such that $M|F$ is connected.

Examples of matroids include *graphic matroids* and *vector matroids*. The former are derived from graphs in the following way: their elements are edges and a set of edges is independent if it does not span a cycle. Vector matroids have vectors as their elements and a set of vectors is independent if the vectors in the set are linearly independent. A matroid M is called *representable over a field* $\mathbf{F}$ if there exists a vector matroid over $\mathbf{F}$ isomorphic to M. A matroid is *binary* if

it is representable over the binary field and it is *regular* if it is representable over any field. Finally, *a uniform matroid* U_n^r is a matroid defined on the universe of size n where independent sets are precisely those with at most r elements.

When a matroid is given as the input of an algorithm by an oracle (e.g., by an oracle encoding the set system $\mathcal{I}$), the number of elements of the matroid is used instead of the input length when discussing the time complexity. Furthermore, the time the oracle spends computing the answer is not counted towards the number of steps the main algorithm took – only the time spent on constructing the input for the oracle and reading its output is accounted for in the overall runtime.

2.11 Width parameters for matroids

A branch decomposition of a matroid $M = (E, \mathcal{I})$ corresponds to how the matroid M might be constructed by "gluing" elementary matroids along separations of small size. The decomposition is an unrooted tree T in which all inner nodes have degree exactly 3 and the leaves of T are in one-to-one correspondence with the elements of E. Let us consider an edge e of T and define E_1 and E_2 as the subsets of $E(M)$ corresponding to the leaves of the two components of $T \setminus e$. Then *the width of an edge* e *of* T is defined as $\mathrm{r}(E_1) + \mathrm{r}(E_2) - \mathrm{r}(E) + 1$ (note that (E_1, E_2) is a separation in M and the width of e is equal to its size). *The width of the branch decomposition* T is defined as the maximum width of an edge $e \in T$. *The branch-width* $\mathrm{bw}(M)$ *of a matroid* is the least value k for which a branch decomposition of M with width k exists.

The question of constructing a branch decomposition of a small width was

Figure 2.1: The underlying graphs of a pair of matroids (left) and the underlying graph of the graphic matroid $M_1 \odot_{p_1,p_2} M_2$ (right), where M_1, M_2 is the pair of matroids shown in the left part of the figure and p_1, p_2 are the edges represented by dashed lines.

settled in [69, 70] for general matroids (given by an oracle).

Theorem 18 (Corollary 7.2, [69])**.** *For each k, there is an $\mathcal{O}(|E|^4)$ algorithm constructing a decomposition of width at most $3k-1$ or concluding that the matroid has branch-width at least $k+1$.*

Moreover, for matroids representable over a fixed finite field, an efficient algorithm for constructing a branch decomposition of optimal width is given in [43].

Let M_1 and M_2 be two matroids satisfying $p_i \in E(M_i)$, for $i \in \{1,2\}$. Then, the 2-sum $M_1 \odot_{p_1,p_2} M_2$ is defined as the matroid with the set of circuits below:

$$\mathcal{C} = \mathcal{C}(M_1 \setminus p_1) \cup \mathcal{C}(M_2 \setminus p_2) \cup$$
$$\{(C_1 \setminus p_1) \cup (C_2 \setminus p_2) : p_i \in C_i \in \mathcal{C}(M_i) \text{ for } i \in \{1,2\}\}.$$

An example of a 2-sum of a pair of graphic matroids can be found in Figure 2.1.

A monadic second-order (MSO) formula ψ for a matroid M can contain the following:

- logical connectives $\vee, \wedge, \neg, \Rightarrow$,
- the equality predicate $=$,

- quantifications $\exists x$ over elements of $E(M)$ – in this case, we call x *an element variable*,
- quantifications $\exists X$ over subsets of $E(M)$ – there, X is called *a set variable*,
- the predicate $x \in X$ of containment of an element in a set,
- and, finally, the independence predicate $\mathrm{ind}(X)$ determining whether a subset X of $E(M)$ is independent.

In line with the above, lowercase letters (such as $x_1, x_2, \ldots$) are used to denote element variables while uppercase letters (such as $X_1, X_2, \ldots$) denote set variables.

Deciding MSO properties of matroids is NP-hard in general, since, for example, the property that a graph is hamiltonian can be determined by deciding the following formula on the graphic matroid corresponding to the input graph:

$$\exists H \exists e \big(\mathrm{is\_circuit}(H) \wedge \mathrm{is\_base}(H \setminus \{e\})\big),$$

where H is a set variable, e an element variable, and $\mathrm{is\_circuit}(\cdot)$ and $\mathrm{is\_base}(\cdot)$ are predicates testing the property of being a circuit and a base, respectively. These can be defined in MSO logic as follows:

$$\mathrm{is\_circuit}(H) \equiv \big(\neg \mathrm{ind}(H)\big) \wedge \big(\forall e : (e \in H) \Rightarrow \mathrm{ind}(H \setminus \{e\})\big),$$
$$\mathrm{is\_base}(H) \equiv \neg\big(\exists e : \mathrm{ind}(H \cup \{e\})\big).$$

Chapter 3

Permutation Pattern Matching

In this chapter, we study an algorithmic problem where we are given two permutations σ and π and are interested in whether σ is a pattern of π. A permutation π contains a permutation σ as a pattern if it contains a subsequence of length $|\sigma|$ whose elements are in the same relative order as in the permutation σ. This is illustrated in Figure 3.1.

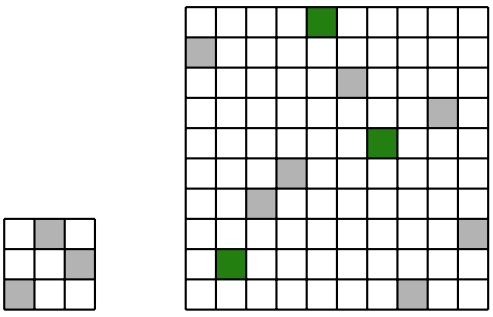

Figure 3.1: A representation of the permutation matrices of permutations $(1, 3, 2)$ *and* $(9, 2, 4, 5, 10, 8, 6, 1, 7, 3)$*. White positions of the grid correspond to 0-entries of the permutation matrix, non-white positions to 1-entries, the columns are indexed from left to right, the rows from bottom to top. Thus, the* $(1, 1)$ *entry of both matrices is located in the bottom-left corner. The former permutation is contained within the latter as a pattern. One of the occurrences of the pattern is highlighted in green.*

The properties of such a partial order on the set of all permutations have been investigated from a variety of angles in discrete mathematics, particularly

in enumerative combinatorics. Knuth [55] has shown that the number of permutations avoiding $(2, 3, 1)$ is the n^{th} Catalan number. Various choices of prohibited patterns have been studied among others by Lovász [60], Rotem [77], and Simion and Schmid [80]. This culminated in the Stanley-Wilf conjecture stating that for every fixed prohibited pattern, the number of permutations of length n avoiding it can be bounded by c^n for some constant c. Klazar [54] reduced the question to the Füredi-Hajnal conjecture, which was ultimately proved by Marcus and Tardos in 2004 [65].

Wilf [83] also asked the algorithmic question of whether detecting a given pattern (of length ℓ) in a given permutation (of length n) can be done in subexponential time. Subsequently, the problem was shown to be NP-hard in [12]. Ahal and Rabinovich have obtained an $\mathcal{O}(n^{0.47\ell+o(\ell)})$ time algorithm [1]. Fast algorithms have been found for certain restricted versions of the problem [12,48]. The linear time dynamic programming algorithm for finding the longest increasing subsequence [78] is even a standard content of many undergraduate courses on algorithms.

Pattern matching has also received interest in the context of parameterized complexity. Several groups of researchers have obtained W[1]-hardness results for generalizations of the problem [15, 39]. For example, in one such generalization the input permutations are colored and the requirement is to find a color-preserving occurrence of the pattern. In [14] it was shown that the problem is in FPT when parameterized by the number of runs (maximal monotonic consecutive subsequences) in the target permutation. The authors of [14] raise the issue of whether their problem has a polynomial size kernel as an open problem. The central question of whether the problem is in FPT when parameterized by ℓ

has been resolved by Guillemot and Marx [39], who obtained an algorithm with asymptotic running time of $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n$. This implies the existence of a kernel for the problem. Obtaining kernel size lower bounds was posed as an open question during a plenary talk at *Permutation Patterns 2013* by Stéphane Vialette.

In what follows, we prove that the permutation pattern problem under the standard parameterization by ℓ does not have a polynomial size kernel (assuming $\mathsf{NP} \not\subseteq \text{co-}\mathsf{NP}/\text{poly}$). This is achieved by introducing a novel polynomial reduction from the Clique problem to Permutation Pattern Matching and applying the cross-composition machinery described in Section 2.9.

3.1 Problem definition and additional notation

Definition 19. *A permutation σ on the set $[l]$ is* a pattern of a permutation π *on the set $[n]$ if there exists an increasing function $\varphi : [l] \to [n]$ such that*

$$\forall x, y \in [l] : \sigma(x) < \sigma(y) \text{ if and only if } \pi(\varphi(x)) < \pi(\varphi(y)).$$

We say that the mapping φ certifies the pattern.

Permutation Pattern Matching is the following parameterized algorithmic problem:

Input: a permutation σ on $[\ell]$, a permutation π on $[n]$.
Parameter: ℓ.
Question: is σ a pattern of π?

In this scenario, the permutation σ is called *the pattern permutation* while π is

the target permutation.

In the sections below, we use the following additional notation. Recall there are two common representations of a permutation π are used in the thesis: using the vector $(\pi(1), \pi(2), \ldots, \pi(n))$ and using a permutation matrix. Both are illustrated in Figure 3.1. A vector obtained from the vector representation by omitting some entries is *a subsequence of the permutation.* Such subsequence is *a consecutive subsequence* if it contains precisely the entries with indexes from $[i, j]$ for some $i, j \in \mathbf{N}$. We use $\pi[i, j]$ to denote the set of entries $\{\pi(i), \pi(i+1), \ldots, \pi(j)\}$. *A monotonic subsequence* is a subsequence whose entries form a monotonic sequence. *A run* is a maximal monotonic consecutive subsequence. For example, $(4, 5, 3, 1, 2)$ contains a (decreasing) run of length 3.

3.2 Kernelization lower bounds

The main result of this thesis is the following theorem, which has already been stated in Introduction.

Theorem 2. *Unless* $\mathsf{NP} \subseteq$ *co-*NP*/poly, the* Permutation Pattern Matching *problem does not have a polynomial kernel.*

We prove Theorem 2 using Theorem 17. However, this requires a polynomial time reduction that allows cross-composition without significantly increasing the parameter value. Reductions described in the literature [12, 15] have resisted our attempts to apply the framework. Therefore, we introduce a new NP-hardness proof that directly leads to a cross-composition. The new reduction is from the well known Clique problem.

Let us first introduce an encoding $\pi_z(G)$ taking a graph G and $z \in \mathbf{N}$ and

producing a permutation. The key property of the encoding is that for any clique K_l on l vertices and any graph H we have $K_l \subseteq H$ if and only if $\pi_z(K_l)$ is a pattern of $\pi_z(H)$ for some particular choice of z. (The value of z depends only on the size of the largest connected component of H.) This allows us to express the Clique problem in terms of Permutation Pattern Matching.

The definition of $\pi(\cdot)$ is somewhat technical although the basic idea is quite simple: we embed the upper-triangular submatrix of the adjacency matrix of the input graph into a permutation. This is illustrated in Figure 3.2, where one can see the permutation matrix of one such encoding permutation. In what follows, we first give a rough sketch of how the construction of the encoding is organized. Then, we describe the individual parts of the resulting permutation in more detail and introduce some notation. Finally, the precise definition is given.

The encoding permutation itself consists of two types of entries: encoding entries and separating entries. The former encode the edges of G. The encoding entries of the same vertex form a consecutive subsequence of the permutation. The separating entries form decreasing runs used to separate encoding entries of different vertices. Looking at $\pi_z(G)$ as an embedding of the upper-triangular submatrix of the adjacency matrix of G offers another perspective: the separating runs mark where each row and column begins and ends, the encoding entries determine where the 1-entries of the matrix are.

We start constructing $\pi_z(G)$ by imposing a total order on $V(G)$ placing vertices from the same connected component of G consecutively. Thus, we can

assume that $V(G) = [n]$ and set

$$N_G^+(v) := \{u : u > v \wedge \{u, v\} \in E(G)\},$$
$$N_G^-(v) := \{u : u < v \wedge \{u, v\} \in E(G)\},$$
$$\deg_G^+(v) := |N_G^+(v)|,$$
$$\deg_G^-(v) := |N_G^-(v)|.$$

The vertices from the sets $N_G^+(v)$ and $N_G^-(v)$ are called *the right-neighbours* and *left-neighbours of* v, respectively. We now give a general overview of the structure of the permutation $\pi_z(G)$; the specification of the exact values and indexes employed is postponed to the following paragraphs. The permutation starts with a decreasing run of length z, continues with the entries encoding the vertex 1 (i.e., encoding $N_G^+(1)$), which is then followed by another decreasing run of length z. This finishes the part of the permutation dedicated to the vertex 1 and the segment for the vertex 2 begins. Again, it starts with another decreasing run of length z, continues with the encoding entries of $N_G^+(2)$, and is finished by a decreasing run of length z. This continues for all vertices of G. Note that for each vertex v there is a pair of decreasing runs immediately surrounding the entries encoding $N_G^+(v)$, one from left and one from right. These are called *the left and right separating runs of* v, respectively. Together, we call these entries *the pair of separating runs of* v. For example, four pairs of separating runs are depicted in Figure 3.2.

To facilitate the formal definition of $\pi_z(G)$, we begin by introducing a notation for important positions and values of the resulting permutation's entries. This includes the positions where the abovementioned runs start, the values with which

they start, or the positions where the parts encoding $N_G^+(v)$, for individual choices of v, start.

We use $p_L(v)$ and $p_R(v)$ as a shorthand for the positions on which the left and right separating run of v starts, respectively. The first position of the segment encoding $N_G^+(v)$ is denoted by $p_M(v)$. This is illustrated in Figure 3.2. Specifically, we set $p_L(1) := 1, p_M(1) := z + 1$, and $p_R(1) := z + 1 + \deg_G^+(1)$.

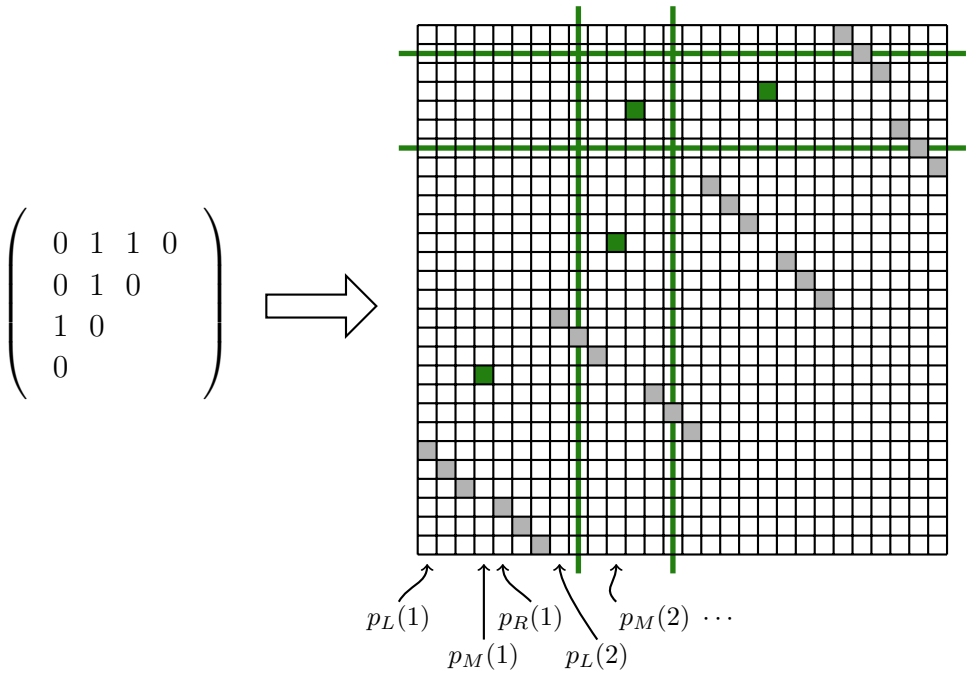

Figure 3.2: Left part shows the upper triangular submatrix of the adjacency matrix of a graph $G = \big(\{1,2,3,4\}, \big\{\{1,2\},\{2,3\},\{2,4\},\{3,4\}\big\}\big)$. *The right part shows the permutation matrix representation of its encoding permutation* $\pi_3(G)$. *Once again, the columns of both matrices are indexed from left to right, the rows from bottom to top. White positions of the grid on the right correspond to 0-entries of the permutation matrix, non-white positions are 1-entries. Separating runs are colored in light gray, encoding entries in dark green. Note the one-to-one correspondence between the 1-entries of the matrix on the left with the encoding entries of the permutation matrix. Horizontal green lines represent values attained at positions* $L(4)$ *and* $R(4)$. *Vertical green lines denote indexes* $L(2)$ *and* $R(2)$. *Note that these four green lines induce a rectangle with a single 1-entry. This encodes the 1-entry in the top-most position of the second column of the adjacency matrix. Arrows below the permutation matrix illustrate the notation* $p_L(\cdot), p_M(\cdot)$, *and* $p_R(\cdot)$.

For $v \geq 2$, we have:

$$p_L(v) := p_R(v-1) + z,$$
$$p_M(v) := p_L(v) + z,$$
$$p_R(v) := p_M(v) + \deg^+_G(v).$$

We also introduce notation for the *values* used by the separating runs. The left separating run of v starts at the position $p_L(v)$ with the value $q_L(v)$. The right separating run starts at $p_R(v)$ with the value $q_R(v)$. Finally, $q_M(v)$ is the least value used by the encoding entries of vertices from $N^-_G(v)$ to determine their connection to v. (Specifically, vertices of $N^-_G(v)$ use the values $[q_M(v), q_M(v) + \deg^-_G(v) - 1]$ to encode this. If $\deg^-_G(v)$ is zero, the value $q_M(v)$ is actually not used.) We set $q_L(1) := 2z, q_M(1) := z + 1$, and $q_R(1) := z$. For $v \geq 2$, let

$$q_R(v) := q_L(v-1) + z,$$
$$q_M(v) := q_R(v) + 1,$$
$$q_L(v) := q_M(v) + z + \deg^-_G(v) - 1.$$

We now define the values of $\pi = \pi_z(G)$. For each v, we introduce a decreasing run of length z starting at the position $p_L(v)$:

$$\pi(p_L(v)) := q_L(v),$$

$$\pi(p_L(v)+1) := q_L(v)-1,$$

$$\pi(p_L(v)+2) := q_L(v)-2,$$

$$\dots$$

$$\pi(p_L(v)+z-1) := q_L(v)-(z-1).$$

We also insert a decreasing run which starts at the position $p_R(v)$ with the value $q_R(v)$:

$$\pi(p_R(v)) := q_R(v),$$

$$\pi(p_R(v)+1) := q_R(v)-1,$$

$$\pi(p_R(v)+2) := q_R(v)-2,$$

$$\dots$$

$$\pi(p_R(v)+z-1) := q_R(v)-(z-1).$$

This describes the entries represented by gray squares in Figure 3.2.

The remaining values are used to encode the edges of G. The neighbourhood $N_G^+(v)$ is encoded by an increasing run on positions $p_M(v), p_M(v)+1, \dots, p_M(v)+|N_G^+(v)|-1$. We fix a vertex $v \in V(G)$ and iterate through the neighbours $\{u_1, u_2, \dots, u_k\} = N_G^+(v)$. Assume $u_1 < u_2 < \dots < u_k$. For $i \in [k]$, we set:

$$\pi(p_M(v)+i-1) := q_M(u_i) + \ell(v, u_i), \tag{3.1}$$

where $\ell(v, u_i) = |\{w : w < v \wedge \{w, u_i\} \in E(G)\}|$. The term $\ell(v, u_i)$ ensures that no value in π is repeated.

The above procedure is carried out for each $v \in V(G)$. This finishes the construction of $\pi_z(G)$. We now provide two observations.

Observation 20. *For any graph G and $z \in \mathbf{N}$ the function $\pi_z(G)$ is a permutation.*

Proof. Let $\pi := \pi_z(G)$. It is straightforward to verify that π is a mapping from $[p]$ to $[p]$, for $p = 2zn + |E(G)|$. It remains to show that π is injective, i.e. that there is no pair of distinct indexes i, j such that $\pi(i) = \pi(j)$. It can easily be seen that such i and j cannot both be an index of an entry forming a separating run, since the separating runs are explicitly constructed so that the sets of their values are disjoint. For each vertex v there are exactly $\deg^-_G(v)$ values between the values of its left and right separating run. These values are used to encode the $\deg^-_G(v)$ edges connecting v to its left-neighbours. The left-most neighbour is using the least value, the subsequent vertices are using values that increase by 1 with each neighbour (cf. the term $\ell(v, u_i)$ in equation (3.1)). Therefore, we have neither a collision between a separating entry and an encoding entry nor a collision between two entries encoding $N^-_G(v)$ for the same v. Finally, it can be easily seen that the sets of values encoding $N^-_G(v)$ are pairwise disjoint for different choices of v. □

Observation 21. *For any choice of $z \in \mathbf{N}$ and any choice of $u, v \in V(G)$, there is at most one 1-entry of $\pi_z(G)$ with an index in $[p_M(u), p_R(u) - 1]$ and value in $[q_M(v), q_M(v) + \deg^-_G(v) - 1]$. Furthermore, there is an edge between vertices $u, v \in V(G), u < v$ if and only if there is exactly one such 1-entry.*

Proof. The entries of $\pi := \pi_z(G)$ with indexes from $[p_M(u), p_R(u)-1]$ encode the neighbourhood of the vertex u, i.e., $N_G^+(u)$. For each neighbour v from $N_G^+(u)$, we insert a single entry with value from $[q_M(v), q_M(v) + \deg_G^-(v) - 1]$. This is because in equation (3.1) the term $\ell(v, u_i)$ is always strictly less than $\deg_G^-(v)$. This implies both parts of the observation. □

For the purpose of the proof of the lemma below, we define the following:

$$C(v) := [p_M(v), p_R(v) - 1],$$

$$L(v) := p_L(v) + \lfloor \tfrac{z}{2} \rfloor,$$

$$R(v) := p_R(v) + \lfloor \tfrac{z}{2} \rfloor.$$

Therefore, $C(v)$ is the set of entries of π encoding the vertex v, $L(v)$ denotes the middle entry of the left separating run of v, and $R(v)$ is the middle entry of the right separating run of v. Once more, Figure 3.2 illustrates the notation.

The following lemma implies NP-hardness of the studied problem:

Lemma 22. *For every clique G, K_l is a subgraph of G if and only if $\pi_z(K_l)$ is a pattern of $\pi_z(G)$, for $z = 4n' + 4$, where n' is the number of vertices in the largest connected component of G.*

Proof. We let $\sigma := \pi_z(K_l)$ and $\pi := \pi_z(G)$.

If G contains a clique K_l of size l as a subgraph, then π contains the pattern σ by construction. This is because if we consider the permutation matrix representation of π and delete all columns except the ones that correspond to separating and encoding entries for vertices of $K_l \subseteq G$, we get a matrix that differs from the permutation matrix of σ only by the additional presence of columns that encode

the connection of the vertices of K_l to the vertices outside of K_l. By deleting these columns (and the empty rows resulting from the above deletions) we arrive at the permutation matrix representation of σ, implying σ is a pattern of π.

For the other direction, assume there is a function $\varphi : [|\sigma|] \to [|\pi|]$ certifying the pattern. We start by noting that there are no decreasing subsequences of length $\frac{1}{4}z$ in π avoiding all separating runs. This is because such a sequence contains at most one entry from $C(v)$ for each $v \in V(G)$. At the same time, it cannot simultaneously contain an entry from $C(u)$ and $C(v)$ for u, v chosen from different connected components. This is because the construction of the encoding permutation places vertices from the same component consecutively and the entries encoding a component placed earlier in the ordering have strictly smaller values than those from a later component. This bounds the length of the subsequence by $n' < \frac{1}{4}z$.

Furthermore, *any* decreasing subsequence of π contains entries from at most one pair of separating runs. This is because once the sequence hits a separating run of a vertex v, all its subsequent entries can only be from the pair of separating runs of v. Any decreasing subsequence therefore starts with less than $\frac{1}{4}z$ encoding entries, which are then followed by entries of a pair of separating runs of some vertex.

We now show that the certifying function φ naturally leads to a mapping from $V(K_l)$ to $V(G)$. Consider any vertex $v \in V(K_l)$. The function φ maps the subsequence of σ formed by the pair of separating runs of v to a decreasing subsequence of π of the same length. As argued, such a long decreasing subsequence starts with less than $\frac{1}{4}z$ encoding entries of π, which are then followed by at least $\frac{7}{4}z$ entries from a pair of separating runs of some vertex $u \in V(G)$. This

implies that the middle entry $L(v)$ of the left separating run of v in σ needs to be mapped by φ to the left separating run of $u \in G$. Additionally, the middle entry $R(v)$ of the right separating run of v in σ needs to be mapped by φ somewhere in the right separating run of the same vertex u. The above establishes a mapping from $V(K_l)$ to $V(G)$ denoted by f_φ.

We claim f_φ to be a graph homomorphism. Fix any pair of vertices v_1, v_2 of K_l such that $v_1 < v_2$. We show that $f_\varphi(v_1), f_\varphi(v_2)$ are connected by an edge in G. Since there is an edge between v_1 and v_2 in K_l, Observation 21 implies that the set of values $\sigma[L(v_1), R(v_1)]$ contains precisely one number p with $\sigma(R(v_2)) \leq p \leq \sigma(L(v_2))$. Since φ certifies the pattern σ in π, there needs to be an entry of π with an index between $\varphi(L(v_1))$ and $\varphi(R(v_1))$ and value between $\pi(\varphi(R(v_2)))$ and $\pi(\varphi(L(v_2)))$. Observation 21 then implies there is an edge between $f_\varphi(v_1)$ and $f_\varphi(v_2)$. Thus, f_φ is a homomorphism and G contains a clique of size l. □

The above reduction can be directly used within the cross-composition framework to show our result.

Proof of Theorem 2. We set L to be the set of all pairs (K_l, G), where K_l is a clique, G is a connected graph containing K_l as a subgraph. It is widely known that deciding $x \in L$ is NP-complete.

We introduce a cross-composition of L into Permutation Pattern Matching. Let R be an equivalence relation on $\{0,1\}^*$ with the following properties: the binary sequences that are not representing a pair (K_l, G), where K_l is a clique and G a graph, are placed in a single equivalence class designated for malformed input sequences; a pair of strings representing instances (K^1, G^1) and (K^2, G^2),

respectively, is related in R if and only if $|V(K^1)| = |V(K^2)|, |V(G^1)| = |V(G^2)|$. Clearly, R is a polynomial equivalence relation. For instances (K_l, G_1), (K_l, G_2), (K_l, G_3), ..., (K_l, G_t) from the same equivalence class of R, we produce an instance of the Permutation Pattern Matching problem where we ask if $\pi_z(K_l)$ is in $\pi_z(G)$, where G is a disjoint union of graphs $G_1, \ldots, G_t$ and z is set to $4 \cdot |V(G_1)| + 4$. Lemma 22 shows that the answer to this problem is YES if and only if at least one of the instances (K_l, G_i) belongs to L. Since the parameter of the pattern matching instance is $|\pi_z(K_l)|$, which can be bounded by $|V(G_i)| \cdot 2z + |V(G_i)|^2$ for any i, we can apply Theorem 17. □

3.3 Conclusion

Guillemot and Marx [39] have shown that the Permutation Pattern Matching problem can be solved in $2^{\mathcal{O}(\ell^2 \log \ell)} \cdot n$ time. They raised the question of whether a faster FPT algorithm could be obtained and outlined a strategy for achieving this using their notion of decompositions of permutations. This relied on the bound from the Stanley-Wilf conjecture not being tight. However, Fox [36] has shown the bound is actually tight for almost all permutations. (Still, Fox [36] gives an improved $2^{\mathcal{O}(\ell^2)} \cdot n$ algorithm.)

Note that in order to rule out kernels of size $P(n)$ for *any* polynomial $P(\cdot)$, it suffices to find a cross-composition satisfying *some* polynomial constraint on the value of the parameter of the resulting instance. In particular, the strength of our bound does not depend on how slowly the value of the parameter of the resulting instance grows. Our result rules out all polynomial upper bounds on the kernel size of the Permutation Pattern Matching problem.

Chapter 4

Optimum Linear Arrangement

In the present chapter we give subexponential computational complexity lower bounds for the Optimum Linear Arrangement problem relative to the Min Bisection problem on d-regular graphs. This is achieved by introducing a polynomial reduction from (a variant of) the Min Bisection problem to the Optimum Linear Arrangement problem. The key property of this reduction, which allows us to prove our bounds, is that the instances of the former problem result in instances of Optimum Linear Arrangement with at most $\mathcal{O}(nd)$ edges. Therefore, the size of the instance is increased only linearly.

This reduction employs expanders in a black-box manner. We use the algorithm implicit in Theorem 12 to construct these expanders. Other efficient constructions could be substituted. Actually, any efficient construction generating d-regular graphs G with Cheeger number $h(G) \geq \frac{d}{C}$ for some fixed constant $C \in \mathbf{N}$ would suffice – potentially at the cost of an increase in multiplicative constants irrelevant for the main hardness result. Let us note that this is one of the aspects in which the contents of this chapter differ from the treatment in the corresponding conference paper [6]. In this thesis, we assume the expander

construction algorithm generates graphs with $h(G) \geq \frac{d}{3}$. In the conference paper, the dependency on the value of the Cheeger number is made explicit at the cost of having to deal with more complicated formulas. However, all ideas behind the argument remain identical.

We begin by formalizing the abovementioned notions. The Min Bisection problem is a classic NP-hard problem studied in various different areas, including approximation algorithms [30,31,53,72], parameterized complexity [24], heuristic algorithms [16,17], and average case complexity [18]:

Min Bisection

Input: A graph $G = (V, E)$ on n vertices, an integer k.

Question: Is there a balanced bipartition (A, B) of G with $|E(A, B)| \leq k$?

We focus on a variant that differs in two regards. Firstly, we restrict the input to graphs that are d-regular for some fixed choice of $d \in \mathbf{N}$. Secondly, we consider a gap version of the problem. This variant is referred to as the (d, α, β)-Gap Min Bisection problem, where $d \in \mathbf{N}$ and $0 \leq \alpha \leq \beta \leq 1$.

(d, α, β)-Gap Min Bisection

Input: A d-regular graph $G = (V, E)$ on n vertices.

Output: If there exists a balanced bipartition (A, B) of G such that $|E(A, B)| \leq \frac{\alpha}{2}dn$, output "YES". If there is no balanced bipartition (A, B) of G such that $|E(A, B)| \leq \frac{\beta}{2}dn$, output "NO". Otherwise, the output can be arbitrary.

To introduce the Optimum Linear Arrangement problem, we need some definitions. *A linear arrangement of a graph* G is an injective mapping π :

$V(G) \to \{1, \ldots, n\}$. *The value of the linear arrangement* π is $\sum_{\{u,v\} \in E(G)} |\pi(u) - \pi(v)|$ (the individual summands of this sum are called *the costs of the edges* (u, v) *in* π). We call π *the optimum linear arrangement* if its value is minimized and denote this value by $\mathrm{OLA}(G)$. The Optimum Linear Arrangement problem (also OLA) is defined as follows.

Optimum Linear Arrangement (OLA)
Input: A graph $G = (V, E)$, an integer k.
Question: Does there exist a linear arrangement π of G with value at most k?

Our reduction allows one to solve the instances of (d, α, β)-Gap Min Bisection by solving the resulting instance of Optimum Linear Arrangement. This relates OLA to the following conjecture:

Conjecture 3. *There exist* $d_0 \in \mathbf{N}$ *and* $\alpha, \beta \in (0, 1), \alpha < \beta$ *such that for each* $d \geq d_0$ *there is no* $2^{o(n)}$ *time algorithm for* (d, α, β)-Gap Min Bisection.

The main result of this chapter is:

Theorem 4. *Unless Conjecture 3 fails, there is no* $2^{o(n+m)}$ *time algorithm for* Optimum Linear Arrangement*, where* n *is the number of vertices of the input graph and* m *the number of its edges.*

Even though Conjecture 3 is not one of the widely believed hypotheses of the theory of computational complexity, we believe it to be a reasonable starting point to derive lower bounds such as the one from Theorem 4. We are, however, not aware of any provably equivalent conjectures employed in the literature. Some reasons for this have already been mentioned in Chapter 1. Additionally, the

best known approximation algorithm for the Min Bisection problem has an approximation ratio $\mathcal{O}(\log n)$ [72]. In order for such an algorithm to violate Conjecture 3, an approximation ratio arbitrarily close to 1 would be necessary.

We proceed by first describing a reduction $T(\cdot)$ from an instance G of Min Bisection to an instance of Optimum Linear Arrangement. A technical lemma, called *Swapping Lemma*, is proven. This is used to prove several other lemmas that shed light on the structure of the optimum linear arrangement of the instance $T(G)$. Finally, we prove Theorem 4 by showing how to decide the instances of (d, α, β)-Gap Min Bisection based on the value of the optimum linear arrangement of the instances obtained after applying the reduction $T(\cdot)$. Note that the values α and β are not under our control. The parameters of our reduction will depend on the gap between them, with smaller gaps leading to bigger multiplicative factors in the size of the resulting instances. Thankfully, the conjecture guarantees that the choice of α and β is fixed for all problem instances. The fact that our reduction exhibits only a linear increase in the size of the instance is crucial in achieving the $2^{o(m+n)}$ bounds. (This is similar to the proofs of the ETH-based lower bounds reviewed in Section 2.7.)

4.1 Sparse reduction to OLA

The result of transformation $T(\cdot)$ is influenced by several parameters. The choice of their values is deferred to the proof of Theorem 4. Consider an instance G of the Min Bisection problem, where G is a d_G-regular graph. Assume $V(G) = \{v_1, \ldots, v_n\}$. The transformation produces a graph $G' := T(G)$ with the vertex set $\{v_1, \ldots, v_n, x_1, \ldots, x_{Mn}\}$, for some constant $M > 0$ chosen later.

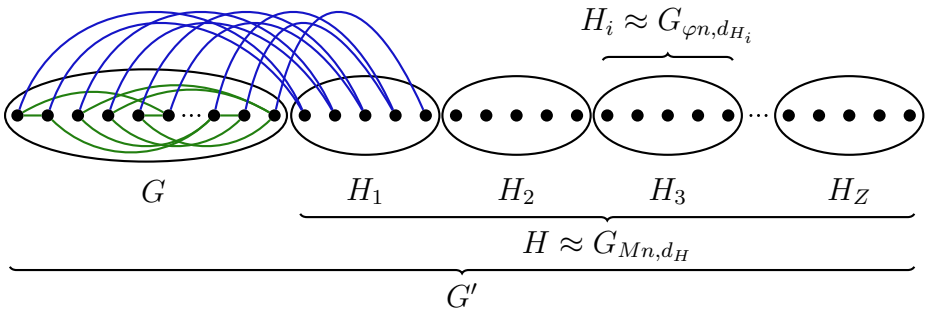

Figure 4.1: The resulting instance $G' = T(G)$ after applying the reduction. The original graph G is an induced subgraph of G' with its edges shown in green. The edges of the bipartite graph added between $V(G)$ and $V(H_1)$ are shown in blue.

Note that G' contains the vertices of G. Indeed, we are going to construct the edge-set in such a way that G is an induced subgraph of G'. It is actually convenient to introduce a notation for some of the induced subgraphs of G'. The subgraph with the vertex set $\{x_1, \ldots, x_{Mn}\}$ is denoted by H. The graph H is (arbitrarily) divided into $Z := \frac{M}{\varphi}$ disjoint induced subgraphs H_i of size φn. We can assume Z and φn to be integers. The result of the transformation is illustrated on Figure 4.1. The edge-set of G' is constructed as follows:

- The induced subgraph of G' on $\{v_1, \ldots, v_n\}$ is G.
- We construct an expander G_{Mn,d_H} using Theorem 12 and add its edges on the vertices of H (recall that G_{Mn,d_H} is a d_H-regular expander on Mn vertices).
- For each $i \in \{1, \ldots, Z\}$ we construct an expander $G_{\varphi n, d_{H_i}}$ using Theorem 12 and add its edges on the vertices of H_i.
- For each $i \in \{1, \ldots, Z\}$ we add a bipartite graph on parts $V(G)$ and $V(H_i)$ such that all vertices of $V(G)$ have degree 1 in this bipartite graph and the degrees of vertices from $V(H_i)$ differ by at most 1. We denote the maximum

degree of the $V(H_i)$ part of this added graph by $\Delta_{H,G}$. It is at most $\lceil \frac{1}{\varphi} \rceil$.

The resulting graph G' is thus influenced (apart from the input graph G) by our choice of parameters M, φ, d_H, and d_{H_i}. In Section 4.2, we give several lemmas that impose a particular structure on the optimum linear arrangement of G', provided certain inequalities between these parameters are satisfied. Eventually, they are employed in the proof of the main theorem of this chapter (Theorem 4), where the values of the parameters are determined. We begin by proving Lemma 23, a technical lemma repeatedly utilized in the next section.

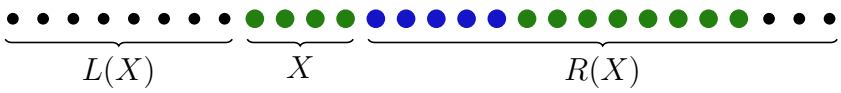

Figure 4.2: For the purposes of our proof, we order the vertices of the resulting graph from left to right. The set of blue vertices is consecutive. The set of green vertices is formed of two blocks. The left-most block is denoted by X. The sets $L(X)$ and $R(X)$ are vertices to the left and right of X, respectively. The blue vertices form the inner block of the green vertices.

In all proofs of this chapter, we understand the vertices of G' to be ordered from left to right according to some ordering σ. When we speak about the i-th vertex (from the left), we mean the vertex mapped to the number i by σ. A set of vertices U is *consecutive in* σ, if $\sigma(U) = \{p, p+1, \ldots, q-1, q\}$ for $p, q \in \mathbf{N}$. This is illustrated in Figure 4.2. The set of vertices that are to the left of every vertex from some set U is called *the set of vertices to the left of* U and denoted by $L(U)$. Similarly, we define *the set of vertices to the right of* U and denote them by $R(U)$. *A block of* U is any inclusion-wise maximal non-empty subset of U that is consecutive in σ. *The left-most block of* U is the block of U whose vertices are mapped to the smallest values by σ. *Second left-most block of* U is the first block of U to the right of the left-most block of U. *Inner block of* U is the set of all

vertices from $V(G') \setminus U$ located simultaneously to the right the left-most block of U and to the left of the second left-most block of U (in the case when U forms a single block, the inner block does not exist).

The following technical *Swapping Lemma* establishes a condition on degrees in two consecutive sets of G' under which the swapping of the two sets results in a decreased cost of the ordering. (Figure 4.3 illustrates the situation.)

Lemma 23 (Swapping Lemma)**.** *Consider an ordering σ of G' obtained by the reduction from a graph G. Assume that the sets $X, Y \subseteq V(G')$ are consecutive and X immediately precedes Y. Let $L := L(X)$ and $R := R(Y)$. Assume*

- *the value P_X upper bounds the degree of vertices from X in the induced bipartite subgraph $G'[L, X]$,*
- *P_C is an upper bound on the maximum degree of $G'[X, Y]$, and*
- *P_Y is an upper bound on the degree of a vertex from Y in $G'[Y, R]$.*
- *Finally, let p be a lower bound on the average degree of a vertex from X in $G'[X, R]$.*

Then the inequality $p > P_X + 2P_C + P_Y$ implies that swapping the vertices of X with the vertices of Y in the order specified by σ results in a decrease in the cost of the ordering.

Proof. The length of edges connecting pairs of vertices from one of the sets L, X, Y, R remains unchanged after swapping X and Y in the ordering. The same holds for edges connecting L with R. The length of each edge connecting X and Y increases by at most $|X| + |Y| \leq 2\max\{|X|, |Y|\}$. The cost of each edge connecting X and L increases by at most $|Y|$. Similarly, the cost of each

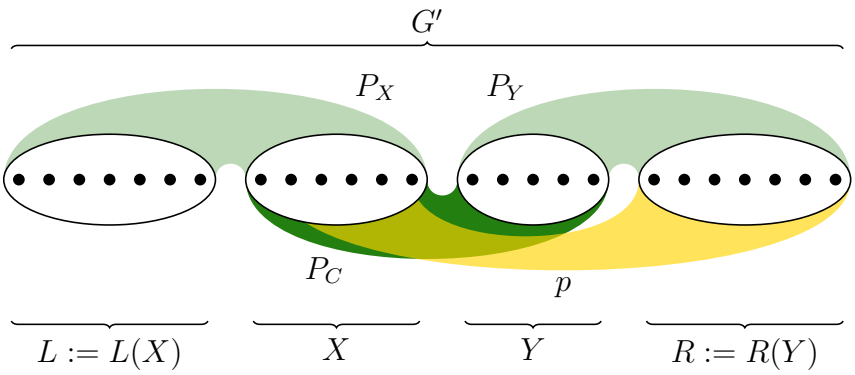

Figure 4.3: The vertex set of G' is partitioned into four sets, L, X, Y, and R, in Lemma 23. The bounds P_X, P_Y, P_C (upper bounds), and p (a lower bound) on the degrees of induced bipartite subgraphs are also shown.

edge connecting Y and R increases by at most $|X|$. On the other hand, the edges connecting X and R are shortened, each by $|Y|$. The upper bounds on the maximum degrees and the lower bound on the average degree from the statement of the lemma allow us to lower-bound the decrease in the cost of ordering after the swap is performed. For example, the decrease in total cost of the edges connecting X with R is at least $p|X||Y|$. The decrease in cost after swapping is at least

$$p|X||Y| - 2\min\{|X|,|Y|\}P_C \max\{|X|,|Y|\} - |X|P_X|Y| - |Y|P_Y|X|,$$

which is equal to

$$|X||Y|(p - 2P_C - P_X - P_Y).$$

Assuming the inequality from the lemma, this is strictly larger than zero. □

4.2 Structure of the solution

We now make several claims about the optimum ordering σ of G'.

Lemma 24. *If $d_H > 9\Delta_{H,G}+9\frac{M}{\varphi}+3d_G$ and σ is the optimum linear arrangement of G', then $V(H)$ is consecutive in σ.*

Proof. Suppose $V(H)$ is not consecutive in σ. Consider the left-most block of $V(H)$ and denote its elements by X. We can assume that $|X| \leq \frac{|H|}{2}$ – otherwise we take the right-most block of $V(H)$ and proceed with a mirrored version of the following argument. Denote by Y the inner block of $V(H)$ and set $L := L(X), R := R(Y)$. The following choice of values satisfies the assumptions on degree upper bounds of Lemma 23:

$$P_X := \Delta_{H,G}, \qquad P_Y := d_G + \frac{M}{\varphi}, \qquad P_C := \Delta_{H,G} + \frac{M}{\varphi}.$$

Since H is an expander and $H \setminus X \subseteq R$, we can take $p = \frac{d_H}{3}$. It remains to show the inequality from the statement of the Swapping Lemma. We have:

$$p = \frac{d_H}{3} > 3\Delta_{H,G} + 3\frac{M}{\varphi} + d_G = P_X + 2P_C + P_Y.$$

Thus we can swap X and Y and decrease the cost of the ordering. This contradicts the optimality of σ. □

Lemma 25. *If $d_{H_i} > 3d_{H_{i+1}}+12d_H+6\Delta_{H,G}$, σ is the optimum linear arrangement of G', and for each $i' < i$ the vertices of $H_{i'}$ are consecutive in σ, then the vertices of H_i are consecutive in σ.*

Proof. Assume H_i not to be consecutive and denote by X the left-most block of $V(H_i)$ in σ. We show that moving X to the right decreases the cost of the arrangement.

Denote by u the vertex positioned by σ immediately to the right of X. Due to

Lemma 24, we know $u \notin V(G)$. Therefore, $u \in V(H_j)$ for $j \neq i$. We distinguish two cases: either $j < i$ or $j > i$.

Suppose that $j < i$. Note that H_j is consecutive in σ. We set $Y := V(H_j), L := L(X)$, and $R := R(Y)$. Again, we employ the Swapping Lemma. The following degree upper bounds satisfy its assumptions:

$$P_X := \Delta_{H,G} + d_H, \qquad P_Y := \Delta_{H,G} + d_H, \qquad P_C := d_H.$$

Since H_i is an expander and $V(H_i) \setminus X \subseteq R$, we can set the average degree lower bound p to $\frac{d_{H_i}}{3}$. By the inequality from the statement of the lemma, we have

$$p = \frac{d_{H_i}}{3} > 4d_H + 2\Delta_{H,G} = P_X + 2P_C + P_Y.$$

Thus, the inequality from the Swapping Lemma holds and we can use it to decrease the cost of ordering, contradicting the optimality of σ.

Now suppose that $j > i$. We now use the Lemma 23 again to move the block X one position to the right, effectively swapping X and $Y := \{u\}$. We set $L := L(X), R := R(Y)$. This time, we set:

$$P_X := \Delta_{H,G} + d_H, \qquad P_Y := \Delta_{H,G} + d_H + d_{H_{i+1}}, \qquad P_C := d_H.$$

Similarly to the previous cases, we set $p := \frac{d_{H_i}}{3}$. The inequality from the Swapping Lemma is again satisfied:

$$p = \frac{d_{H_i}}{3} > 4d_H + 2\Delta_{H,G} + d_{H_{i+1}} = P_X + 2P_C + P_Y.$$

Once more, we get a contradiction with the optimality of σ. □

Due to Lemma 24 we know that the optimal linear arrangement of G' places vertices of H consecutively, assuming the inequalities from its statement are satisfied. By iterating Lemma 25, we get that within H, the vertices from each H_i are grouped together in the optimal arrangement. The vertices of G can thus be only to the left of H or to its right. The next lemma shows that H actually divides the graph G into two roughly equal parts.

Lemma 26. *Let $\gamma > 0$. Assume G' has been constructed by transformation $T(\cdot)$ with parameters satisfying the inequalities from the statements of Lemmas 24 and 25. Further assume that $\varphi = \frac{\gamma}{3d_G}$ and $M \geq 2$. Consider the optimal linear arrangement σ of G' and set $V_1 := L(H), V_2 := R(H)$. Then $\big||V_1| - |V_2|\big| \leq \gamma n$.*

Proof. Assume the imbalance $\big||V_1|-|V_2|\big|$ is strictly larger than γn. Without loss of generality, assume $|V_1| > |V_2|$. We consider the vertex u such that $\sigma(u) = 1$ (i.e., the one placed at the leftmost point in the arrangement).

Moving u to the right-most position results in the following changes in the cost of the arrangement. The cost associated with the edges of G might be increased by at most $d_G(M+1)n$. In addition to this, the vertex u is connected to precisely one vertex of each H_i. We cannot be sure precisely where the neighbouring vertex is mapped by σ, which leads us to a possible increase that is upper-bounded by $\frac{M}{\varphi}\varphi n = Mn$. The possible total cost increase related to this is upper-bounded by $(d_G + 1)(M + 1)n$. However, moving the vertex u to the other side also leads to a decrease of γn for each edge between u and a subgraph H_i. This represents a cost decrease of at least $\gamma n \frac{M}{\varphi}$. In total, we get a cost decrease of $\gamma n \frac{M}{\varphi} - 2d_G Mn = 3d_G nM - 2d_G nM = d_G Mn$. □

For any $\gamma > 0, M \in \mathbf{N}, M \geq 2$, it is possible to satisfy the assumptions of the previous lemmas simultaneously. This allows us to prove the main theorem:

Proof of Theorem 4. The choice of γ and M is deferred to the end of the proof. We set

$$\varphi = \frac{\gamma}{3d_G},$$

which implies $\Delta_{H,G} = \frac{1}{\varphi} = \frac{3d_G}{\gamma}$. We let

$$d_H = 9\Delta_{H,G} + 9\frac{M}{\varphi} + 3d_G + 1,$$

$$d_{H_M} = 12d_H + 6\Delta_{H,G} + 1,$$

$$d_{H_i} = 4d_{H_{i+1}}.$$

If d_H, d_{H_i} are not prime powers (and thus Theorem 12 does not guarantee the existence of an explicit construction of such expander), we increase the value of the parameters to a larger prime power.

We show how to apply our transformation to distinguish instances G of Min Bisection with at most $\frac{\alpha}{2}d_G n$ edges from those with at least $\frac{\beta}{2}d_G n$ edges in the optimum balanced cut. We proceed as follows:

- Denote by G' the result of applying the transformation $T(\cdot)$ to G with parameters γ, M, d_H and $d_{H_i}, i \in \mathbf{N}$.

- Set

$$X := \mathrm{OLA}(H) + \frac{\alpha}{2}d_G n^2(M+1) + \frac{d_G n}{2} \cdot \frac{n}{2} + \frac{(n+2)nM}{4\varphi} + n\sum_{i=1}^{\frac{M}{\varphi}} i\varphi n,$$

  where $\mathrm{OLA}(H)$ is the cost of the optimal arrangement of the induced sub-

graph H from our reduction. Note that if we assume that the graph G has a bisection with at most $\frac{\alpha}{2}d_G n$ edges, then X is an upper bound on the cost of the optimum linear arrangement of G'. This is because it accounts for all costs associated with an ordering of G' constructed from the optimal bisection of G. Denote by A, B the disjoint union of $V(G)$ associated with optimal bisection of G. The abovementioned ordering first lists all vertices of A, then the vertices of H in the order of cost $\mathrm{OLA}(H)$, and then the vertices of B. The first term of X counts the cost of all edges of G' connecting two vertices of H and the second term the cost of edges of G between A and B. The third term the cost of edges within A and within B: there is $\frac{d_G n}{2}$ of these edges and each with a cost of at most $\frac{n}{2}$. The last two terms upper-bound the cost of edges connecting G to H. Every vertex v of G has an edge to exactly one vertex of each H_{ℓ_i}. If $v \in A$, we may bound its cost by $j(v) + i\varphi n$, where $j(v)$ is the length of the part of the ordering from v to the first vertex in H. We first count the contribution of the $j(v)$-terms in the above expression for all choices of $v \in A$. Since $|A| = \frac{n}{2}$, summing over all $v \in A$ and $i = 1, \ldots, Z$ we get

$$\sum_{j=1}^{|A|} j\frac{M}{\varphi} = \frac{\left(\frac{n}{2}+1\right)\frac{n}{2}}{2} \cdot \frac{M}{\varphi} = \frac{1}{2} \cdot \frac{(n+2)nM}{4\varphi}.$$

The situation is analogous for B. The very last term is obtained by summing the remaining edge costs $i\varphi n$ for all $i = 1, \ldots, Z$ and $v \in G$. Clearly, the value of the *optimal* ordering can be only smaller.

- If $\mathrm{OLA}(G') \leq X$, output that the value of Min Bisection of G is at most $\frac{\alpha}{2}d_G n$ (i.e., print "YES").

- Otherwise, we output "NO", corresponding to the statement that the value of Min Bisection of G is at least $\frac{\beta}{2}d_G n$.

It remains to argue there is a choice of constants γ and M, such that it is guaranteed that instances with large optimum bisection always satisfy $\mathrm{OLA}(G') > X$. Suppose there are at least $\frac{\beta}{2}d_G n$ edges in the optimum bisection of G. Lemma 24, Lemma 25, and Lemma 26 together restrict how the optimum ordering of G' can look, which allows us to lower-bound its value. Therefore, the value of the optimal arrangement of G' is at least:

$$\mathrm{OLA}(H) + \Big(\frac{\beta}{2} - \gamma\Big)d_G n^2 M + \frac{(n+2)nM}{4\varphi} + n\sum_{i=1}^{\frac{M}{\varphi}}\big(i-1\big)\varphi n,$$

where H is again the induced subgraph from our reduction. The first term lower-bounds the cost of edges contained in H, the second term the cost of edges in the minimum bisection (with at most $\gamma n d_G$ edges subtracted because we allow for a slight imbalance), and the last two terms again lower-bound the cost of edges connecting $V(G)$ to $V(H)$.

The difference between the lower bound on the value resulting from large instances of Min Bisection and X is at least

$$\frac{(\beta-\alpha-2\gamma)}{2}d_G M n^2 - \frac{\alpha}{2}d_G n^2 - \frac{d_G n}{2}\cdot\frac{n}{2} - n\frac{M}{\varphi}\varphi n,$$

which is easily rewritten as

$$n^2\Big[\frac{(\beta-\alpha-2\gamma)}{2}d_G M - \frac{d_G}{\alpha} - \frac{d_G}{4} - M\Big].$$

This quantity is strictly larger than 0 for $\gamma = \frac{\beta-\alpha}{5}, d_G \geq \frac{10}{\beta-\alpha}$ and sufficiently

large constant M. Thus, we never report a graph with minimum bisection of at least $\frac{\beta}{2} d_G n$ as a small instance. □

4.3 Conclusion

Currently, the Optimum Linear Arrangement problem has a unique position in the theory of NP-hardness because of the following. The only known polynomial reductions proving the hardness of several algorithmic problems (including Chordal Completion, Trivially Perfect Completion, Proper Interval Completion, and Interval Completion) are all routed through OLA. The NP-hardness of OLA itself has been established by a polynomial reduction from Maximum Cut. For the purposes of obtaining subexponential complexity lower bounds, there is a major disadvantage of this reduction: it implies a substantial increase in the instance size. This in turn weakens the resulting lower bounds based on the Exponential Time Hypothesis. We address this in [6]. There, we also use Theorem 4 to obtain bounds on the complexity of the abovementioned problems and their parameterized variants.

Chapter 5

Amalgam-width of matroids

The celebrated theorem of Courcelle [21], presented earlier as Theorem 1, provides a unifying framework for obtaining FPT algorithms for various algorithmic problems defined on graphs parameterized by their tree-width. There are several other width parameters for graphs with similar computational properties, e.g., boolean-width [19] and clique-width [22]. This chapter examines the challenges of extending results of this type to matroids, which are combinatorial structures generalizing the notions of graphs and linear independence. Although the tree-width for matroids has been introduced [44], a more natural width parameter for matroids is branch-width (for its definition, see Section 2.11). This is due to the fact that the branch-width of graphs can be introduced without referring to vertices, which are not explicitly available when working with (graphic) matroids.

However, most of the presently available width parameters, including branch-width, do not allow corresponding extension for general matroids without additional restrictions. Although computing decompositions of nearly optimal width efficiently is usually still possible (the results [69, 70] managed this for branch-width and clique-width), the picture becomes more complicated for de-

ciding properties. Extensions to matroids are feasible but significant obstacles emerge for non-representable matroids. This indicates a need for a width parameter reflecting the complex behavior of matroids that are not representable over finite fields.

Let us be more specific with the description of the state of the art for matroids. The analogue of Courcelle's theorem was proven by Hliněný [42] in the following form:

Theorem 27 (Hliněný, [42])**.** *Let* $\mathbf{F}$ *be a finite field,* φ *be a fixed MSO formula, and* $t \in \mathbf{N}$*. Then there is a linear time algorithm deciding* φ *on* $\mathbf{F}$*-represented matroids parameterized by their branch-width bounded from above by* t*.*

However, as evidenced by several negative results, a full generalization of the above theorem to all matroids is not possible: Seymour [79] has shown that there is no subexponential time algorithm testing whether a matroid (given by an oracle) is representable over GF(2). Note that being representable over GF(2) is equivalent to the non-existence of a uniform U_2^4 minor, which can be expressed in MSO logic. This result generalizes for all finite fields and holds even when restricted to matroids of bounded branch-width. Subsequently, this implies the intractability of deciding MSO formulas on general matroids of bounded branch-width. See [56] for more details on matroid representability from a computational point of view. Besides MSO properties, algorithmic aspects of first-order properties have also been studied [37].

Two width parameters have been proposed to circumvent the restriction of tractability results to matroids representable over finite fields: decomposition-width [57] and another width parameter based on 2-sums of matroids [81]. The latter allows the input matroid to be split only along 2-separations, making it

of little use for 3-connected matroids. On the other hand, though the first one can split the matroid along more complex separations, it does not correspond to any natural "gluing" operation on matroids. In this work, we present a matroid parameter, called amalgam-width, that has neither of these two disadvantages and still allows proving corresponding algorithmic results. An input matroid can be split along complex separations and the parts of the decomposed matroid can be glued together using the so-called amalgamation [71], which is a well-established matroid operation.

5.1 Matroid amalgams

In this section we define the operation of *a generalized parallel connection*, which plays a key role in the definition of an amalgam decomposition. We begin by introducing *matroid amalgams* and *modular flats*.

Definition 28. *Let M_1 and M_2 be two matroids with ground sets E_1 and E_2, respectively. Let $E := E_1 \cup E_2$ and $T := E_1 \cap E_2$. We assume that $M_1|T = M_2|T$. If M is a matroid with the ground set E such that $M|E_1 = M_1$ and $M|E_2 = M_2$, we say that M is* an amalgam *of M_1 and M_2.*

An example of an amalgam of two matroids is given in Figure 5.1, which illustrates several other key notions of this chapter. An amalgam of two matroids does not necessarily exist, even if the matroids coincide on the intersection of their ground sets. Our aim is to investigate a condition on matroids sufficient for the existence of an amalgam. To do so, we introduce the notions of *free amalgams* and *proper amalgams.*

Definition 29. *Let M_0 be an amalgam of M_1 and M_2. We say that M_0 is* the free amalgam *of M_1 and M_2 if for every amalgam M of M_1 and M_2 every set independent in M is also independent in M_0.*

The definition of the more restrictive *proper amalgam* is more involved.

Definition 30. *Let M_1 and M_2 be two matroids with rank functions r_1 and r_2, respectively, and independent sets coinciding on $T := E(M_1) \cap E(M_2)$. First, define functions η and ζ on subsets of $E := E(M_1) \cup E(M_2)$ as follows:*

$$\eta(X) := r_1\big(X \cap E(M_1)\big) + r_2\big(X \cap E(M_2)\big) - r(X \cap T),$$

$$\zeta(X) := \min\{\eta(Y) : Y \supseteq X\},$$

where r is the rank function of the matroid $N := M_1|T = M_2|T$. (Note that η provides an upper bound on the rank of the set X in a supposed amalgam of M_1 and M_2, while ζ is the least of these upper bounds.) If ζ is submodular on $\mathcal{S}(E)$, we say that the matroid on $E(M_1) \cup E(M_2)$ with ζ as its rank function is the proper amalgam *of M_1 and M_2.*

It can be verified that if the proper amalgam of two matroids exists then it is the free amalgam. The next lemma provides a necessary and sufficient condition for an amalgam to be the proper amalgam of two given matroids.

Lemma 31 (Oxley, [71])**.** *Let M_1 and M_2 be two matroids, M one of their amalgams, and T the intersection $E(M_1) \cap E(M_2)$. The matroid M is the proper amalgam of M_1 and M_2 if and only if it holds for every flat F of M that*

$$r(F) = r(F \cap E_1) + r(F \cap E_2) - r(F \cap T).$$

However, Lemma 31 says nothing about the existence of the proper amalgam of M_1 and M_2. Below, we give a condition that guarantees it. The notion of *modular semiflats* is necessary for this.

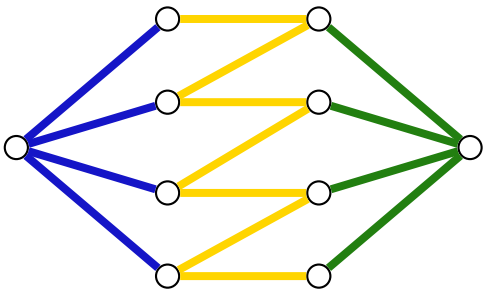

Figure 5.1: Let M_1 be a graphic matroid of a graph that contains exactly the blue and yellow edges from the figure (and the incident vertices). Similarly, let M_2 be a graphic matroid of a graph containing exactly the green and yellow edges. Finally, let M be the graphic matroid of a graph containing all of the above edges. Then, M is an amalgam of M_1 and M_2. Denote by X the yellow edges (which are simultaneously elements of both M_1 and M_2) and Y the set of all blue edges (which are also elements of M_1). Both X and Y are flats of M_1. We have $r(Y) = 4, r(X) = 7, r(X \cup Y) = 10$ and $r(X \cap Y) = 0$. Therefore, X is not a modular flat in M_1. It can be also easily verified that the two matroids violate the statement of Lemma 35. We conclude that the generalized parallel connection of M_1 and M_2 does not exist.

Definition 32. *A flat $X = \mathrm{cl}(T)$ of a matroid M is* modular *if for any flat Y of M the following holds:*

$$r(X \cup Y) = r(X) + r(Y) - r(X \cap Y).$$

Furthermore, we say that T is a modular semiflat *if* $\mathrm{cl}(T)$ *is a modular flat in M and every element of* $\mathrm{cl}(T)$ *is either in T, a loop, or parallel to some other element of T.*

For example, the set of all elements, the set of all loops, and any flat of rank one are modular flats. Each single-element set is a modular semiflat. The following

theorem is given in Oxley's monograph [71]:

Theorem 33. *Suppose that M_1 and M_2 are two matroids with a common restriction $N := M_1|T = M_2|T$, where $T := E(M_1) \cap E(M_2)$. If T is a modular semiflat in M_1, then the proper amalgam of M_1 and M_2 exists.*

We are now ready to introduce the operation of a generalized parallel connection, which can be used to glue matroids.

If M_1 and M_2 satisfy the assumptions of Theorem 33, then the resulting proper amalgam is called *the generalized parallel connection* of M_1 and M_2 and is denoted by $M_1 \oplus_N M_2$, where $N := M_1|\big(E(M_1) \cap E(M_2)\big)$. Since N is uniquely determined by the two matroids, we sometimes omit it and use $M_1 \oplus M_2$ instead. The generalized parallel connection satisfies the following basic properties.

Lemma 34 (Oxley, Proposition 12.4.14(iii), [71])**.** *If the generalized parallel connection of matroids M_1 and M_2 exists, then $\mathrm{cl}\big(E(M_2)\big)$ is a modular semiflat in $M_1 \oplus M_2$.*

Lemma 35 (Oxley, pg. 446, [71])**.** *Let M_1 and M_2 be two matroids on ground sets E_1 and E_2, respectively. Furthermore, let $T := E_1 \cap E_2$, N be the matroid $M_1|T = M_2|T$, and $M := M_1 \oplus_N M_2$. For $X \subseteq E_1 \cup E_2$, let $X_i = \mathrm{cl}_{M_i}(X \cap E_i) \cup X$. It holds that*

$$\mathrm{cl}_M(X) = \mathrm{cl}_{M_1}(X_2 \cap E_1) \cup \mathrm{cl}_{M_2}(X_1 \cap E_2), \text{ and}$$

$$r_M(X) = r_{M_1}(X_2 \cap E_1) + r_{M_2}(X_1 \cap E_2) - r_M\big(T \cap (X_1 \cup X_2)\big).$$

The operation of generalized parallel connection also commutes:

Lemma 36. *Let K, M_1 and M_2 be matroids such that $M_1|T_1 = K|T_1$ and $M_2|T_2 = K|T_2$, where $T_1 := E(M_1) \cap E(K)$ and $T_2 := E(M_2) \cap E(K)$. Let $N_1 := E(M_1) \cap \big(E(M_2) \cup E(K)\big)$ and $N_2 := E(M_2) \cap \big(E(M_1) \cup E(K)\big)$. If T_1 is a modular semiflat in M_1 and T_2 is a modular semiflat in M_2, then*

$$M_2 \oplus_{N_2} (M_1 \oplus_{T_1} K) = M_1 \oplus_{N_1} (M_2 \oplus_{T_2} K).$$

5.2 Amalgam-width

Recall that the class of graphs of bounded tree-width can be introduced as the set of all subgraphs of a k-tree, where a k-tree is a graph that can be obtained by gluing two smaller k-trees along a clique of size k. Similarly, matroids of bounded branch-width can be introduced in terms of an operation taking two matroids of bounded branch-width and producing a larger matroid of bounded branch-width by gluing them along a low-rank separation. The amalgam-width is also defined using a gluing operation. Analogously to the definition of tree-width, where some elements of the clique can be effectively removed after the gluing takes place, the operation includes a set of elements to be deleted. A typical situation when applying the gluing operation is illustrated on Figure 5.2.

Definition 37. *Suppose we are given matroids M_1, M_2, and K such that $E(M_1) \cap E(M_2) \subseteq E(K)$. Furthermore, suppose we are also given a set $D \subseteq E(K)$. Let $J_i := E(M_i) \cap E(K), i \in \{1, 2\}$ and assume the two conditions below hold:*

- *$M_i|J_i = K|J_i, i \in \{1, 2\}$,*
- *J_1 and J_2 are both modular semiflats in K.*

Then, the matroid $M_1 \oplus^{K,D} M_2$ *is defined as follows:*

$$M_1 \oplus^{K,D} M_2 := \big((K \oplus_{J_1} M_1) \oplus_{J_2} M_2\big) \setminus D.$$

We also say that the matroid $M_1 \oplus^{K,D} M_2$ *is a result of gluing of* M_1 *and* M_2 *along* K *and removing the elements of* D*.*

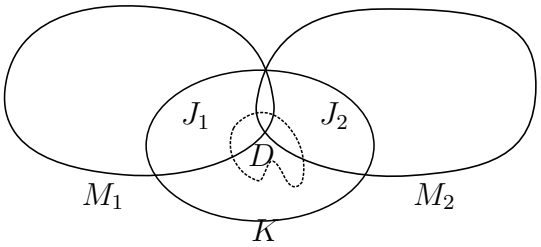

Figure 5.2: M_1, M_2 *are the matroids being combined,* K *is a small matroid used to glue them together, and* D *is a set of elements that are subsequently removed.*

Note that Theorem 33 guarantees the matroid $M_1 \oplus^{K,D} M_2$ to be well defined. We are now ready to introduce our width parameter.

Definition 38. *Matroid* M *has amalgam-width at most* $k \in \mathbf{N}$ *if* $|E(M)| \leq 1$*, or there are matroids* M_1 *and* M_2 *of amalgam-width at most* k*, a matroid* K *satisfying* $|E(K)| \leq k$*, and a choice of* $D \subseteq E(K)$ *such that*

$$M = M_1 \oplus^{K,D} M_2.$$

Note that the first condition can be weakened to $|E(M)| \leq k$ without affecting the definition. Every finite matroid M has an amalgam-width at most $|E(M)|$. The amalgam-width of M is the smallest k such that M has amalgam-width at most k. The definition above naturally yields a tree-like representation of the construction of the matroid in question:

Definition 39. *Assume that M is a matroid with amalgam-width k. Any rooted tree $\mathcal{T}$ satisfying either of the following statements is called* an amalgam decomposition of M of width at most k*:*

- $|E(M)| \leq 1$ *and $\mathcal{T}$ is a trivial tree containing precisely one node,*
- $M = M_1 \oplus^{K,D} M_2$ *and $\mathcal{T}$ has a root r with children r_1 and r_2 such that the subtrees of $\mathcal{T}$ rooted at r_1 and r_2 are amalgam decompositions of M_1 and M_2 of width at most k.*

The above definition leads to a natural assignment of matroids to the nodes of $\mathcal{T}$: whenever a gluing operation is performed, we assign the resulting matroid to the node. We use $M^{\mathcal{T}}(v)$ to refer to this matroid and say that the node v represents $M^{\mathcal{T}}(v)$. For an internal node $v \in \mathcal{T}$, we use $M_1^{\mathcal{T}}(v)$, $M_2^{\mathcal{T}}(v)$, $K^{\mathcal{T}}(v)$, $D^{\mathcal{T}}(v)$, $J_1^{\mathcal{T}}(v)$ and $J_2^{\mathcal{T}}(v)$ to denote the corresponding elements appearing in the gluing operation used to obtain $M^{\mathcal{T}}(v) = M_1^{\mathcal{T}}(v) \oplus^{K^{\mathcal{T}}(v), D^{\mathcal{T}}(v)} M_2^{\mathcal{T}}(v)$. If v is a leaf of a decomposition $\mathcal{T}$, we let $M_1^{\mathcal{T}}(v)$ and $M_2^{\mathcal{T}}(v)$ be matroids with empty ground sets, $K^{\mathcal{T}}(v) = M^{\mathcal{T}}(v)$ the corresponding matroid containing a single element, and $D^{\mathcal{T}}(v) := \emptyset$. Finally, we denote by $J^{\mathcal{T}}(v) \subseteq K(v)$ the set of elements used to glue $M(v)$ to its parent. More formally, we set $J^{\mathcal{T}}(v) := J_i^{\mathcal{T}}(u)$, where $i \in \{1, 2\}$ is chosen depending on whether v is a left or right child of u. Since the decomposition under consideration is typically clear from context, we usually omit the upper index $\mathcal{T}$.

5.3 Algorithms

As the main result of this section, we show that the problem of deciding monadic second-order properties is computationally tractable for matroids of bounded

amalgam-width:

Theorem 7. *Let φ be a fixed formula in monadic second-order logic and $k \in \mathbf{N}$. Then, there is an algorithm deciding whether a matroid M satisfies φ in linear time for graphs of amalgam width bounded from above by k (assuming the corresponding amalgam decomposition $\mathcal{T}$ of the matroid is given explicitly as a part of the input).*

Later, in Section 5.4, we discuss when the requirement of having the amalgam decomposition available as a part of the input can be dropped while maintaining polynomial time complexity. For the purpose of induction used in the proof of Theorem 7, we need to generalize the considered problem by introducing free variables:

INPUT:

- an MSO formula ψ with p free variables,
- amalgam decomposition $\mathcal{T}$ of a matroid M with width at most k,
- a function Q defined on the set $\{1, \ldots, p\}$ assigning the i-th free variable its value; specifically, $Q(i)$ is equal to an element of $E(M)$ if x_i is an element variable, and it is a subset of $E(M)$ if x_i is a set variable.

OUTPUT:

- *ACCEPT* if ψ is satisfied on M with the values prescribed by Q to its free variables,
- *REJECT* otherwise.

The resulting problem is referred to as *the* MSO-DECIDE *problem.* To simplify notation, let us assume that if ψ is a formula with free variables, we use x_i for the i-th variable if it appears in ψ as an element variable and X_i if it appears as a set variable. We prove the following generalization of Theorem 7:

Theorem 40. *For a fixed choice of ψ and $k \in \mathbf{N}$, the problem* MSO-DECIDE *can be solved in linear time (assuming the corresponding amalgam decomposition $\mathcal{T}$ of the matroid is given as a part of the input).*

Our aim in the proof of Theorem 40 is to construct a linear time algorithm based on deterministic bottom-up tree automata. Let us introduce such automata.

Definition 41. A finite tree automaton *is a* 5*-tuple* $(S, S_A, \delta, \Delta, \Sigma)$*, where*

- S *is a finite set of states containing a special initial state* 0*,*
- $S_A \subseteq S$ *is a non-empty set of accepting states,*
- Σ *is a finite alphabet,*
- $\delta : S \times \Sigma \to S$ *is set of transition rules that determine a new state of the automaton based on its current state and the information, represented by* Σ*, contained in the current node of the processed tree, and*
- $\Delta : S \times S \to S$ *is a function combining the states of two children into a new state.*

Let us also establish the following simple notation.

Definition 42. *Consider an instance of an* MSO-DECIDE *problem. In particular, let Q be the variable-assignment function as in the definition of our generalized*

problem. For $F \subseteq E(M)$, we define the local view of Q at F *to be the following function:*

$$Q_F(i) := \begin{cases} Q(i) \cap F & \text{if the } i\text{-th variable is a set variable,} \\ Q(i) & \text{if the } i\text{-th variable is an element variable and } Q(i) \in F, \\ \boxtimes & \text{otherwise,} \end{cases}$$

where $\boxtimes$ is a special symbol that is not an element of the input matroid.

The symbol $\boxtimes$ stands for values outside of F. We simplify the notation by writing $Q_v(x_i)$ instead of $Q_{E(K(v))}(i)$, where v is a node of $\mathcal{T}$ from the problem's input.

The alphabet Σ of the automaton we construct will correspond to the set of all possible "configurations" at a node v in an amalgam decomposition of width at most k. A finite tree automaton processes a tree (in our case $\mathcal{T}$) from its leaves to the root, assigning states to each node based on the information read in the node and on the states of its children. When processing a node whose two children were already processed the automaton calculates the state $s := \Delta(s_1, s_2)$, where s_1 and s_2 are the states of the children, and moves to the state $\delta(s, q)$, where $q \in \Sigma$ represents the information contained in that node of the tree. If the state eventually assigned to the root of the tree is contained in the set S_A, we say that the automaton accepts. It rejects otherwise.

As a final step of our preparation for the proof of Theorem 40, we slightly alter the definition of an MSO formula by replacing the use of $\text{ind}(X)$ predicate with the use of $x_1 \in \text{cl}(X_2)$, where $\text{cl}(\cdot)$ is the closure function of M. The predicate

$\mathrm{ind}(X)$ can be expressed while adhering to the altered definition as follows:

$$\mathrm{ind}(X) \equiv \neg\big(\exists e \in X : \mathrm{cl}(X) = \mathrm{cl}(X \setminus \{e\})\big).$$

Proof of Theorem 40. We proceed by induction on the length of the formula ψ, starting with simple formulas such as $x_1 = x_2$ or $x_1 \in X_2$. In each step of the induction, we design a tree automaton processing the amalgam decomposition tree $\mathcal{T}$ and correctly solving the corresponding MSO-DECIDE problem. As already mentioned, the alphabet will encode all possible non-isomorphic choices of the matroid $K(v)$, sets $J(v), J_1(v), J_2(v)$, and $D(v)$ combined with all possible local views of Q at v, allowing this information to be read when processing the corresponding node. Note that if k is bounded, the size of the set Σ of such configurations is bounded. Since the automaton size does not depend on n and the amount of information read in each node of $\mathcal{T}$ is bounded by a constant (assuming bounded amalgam-width), we will be able to conclude that the running time of our algorithm, which will just simulate the tree automaton, is linear in the size of $\mathcal{T}$.

To start the induction, we first consider the case $\psi \equiv (x_1 \in X_2)$. Such instances of MSO-DECIDE can be solved by the automaton given in Figure 5.3. This automaton stays in its original state if x_1 is assigned $\boxtimes$ by the local view of Q at $E(K(v))$. Otherwise, it moves to designated ACCEPT and REJECT states based on whether $Q_v(x_1) \in Q_v(X_2)$ holds. The set S_A is defined to be $\{\text{ACCEPT}\}$. The function $\Delta : S \times S \to S$ assigns the ACCEPT state to any tuple containing an ACCEPT state. Similarly for the REJECT state. We are guaranteed not to encounter the situation where one child node is in the ACCEPT

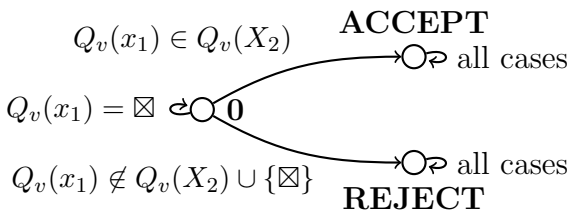

Figure 5.3: The states and transition rules δ of the tree automaton for the formula $x_1 \in X_2$. Here, v is the currently processed node of the amalgam decomposition. The names of the states are typed using bold font.

state and the other in the REJECT state, since the free variable assignment function Q maps x_1 precisely to one element of $E(M)$. It is clear that this tree automaton correctly propagates the information of whether $x_1 \in X_2$ or not from the leaf representing the value of x_1 to the root of $\mathcal{T}$.

The cases of formulas $x_1 = x_2$ and $X_1 = X_2$ can be handled similarly. For formulas of the form $\psi_1 \vee \psi_2$, we construct the automaton by taking the Cartesian product of the automata $A_i = (S^i, S^i_A, \delta^i, \Delta^i, \Sigma^i), i \in \{1, 2\}$ for the partial formulas ψ_i. Specifically, $\Sigma = \Sigma^1 \times \Sigma^2$, $S = S^1 \times S^2$, $S_A = (S^1_A \times S^2) \cup (S^1 \times S^2_A)$, $\Delta\big((x, y)\big) = \big(\Delta^1(x), \Delta^2(y)\big)$, $\delta\big((x, y), (q, r)\big) = \big(\delta^1(x, q), \delta^2(y, r)\big)$. Informally, the two automata run in parallel and the new automaton accepts precisely if at least one of the two is in an accepting state. A formula of the form $\neg\psi$ can be processed by the same automaton as ψ, except we change the set accepting states to their complement. The connectives $\wedge, \Rightarrow, \ldots$ can be expressed using $\vee$ and $\neg$ by a standard reduction.

The special properties of amalgam decompositions come into play when constructing the automaton deciding the formula $x_1 \in \mathrm{cl}(X_2)$. When processing a node $v \in \mathcal{T}$, we see the elements of $K(v)$, can query the independent sets on $E(K(v))$, and see a local view of $Q(X_2)$ at $E(K(v))$. Our strategy will be to compute $\mathrm{cl}_M(X_2)$ restricted to $E(K(v))$ and determine whether x_1 is contained

in it. However, the state at v does not encode necessary information about the remaining part of M. The matroid $M(v)$ is joined to this part by a generalized parallel connection using $J(v)$. Lemma 35 says that the remaining part of M can influence the restriction of the closure of X_2 on $E(K(v))$ only through forcing some of the elements of this modular flat into the closure. Since $|J(v)|$ is bounded, we can pre-compute the behavior of the resulting closure for all possible cases. This information is encoded in the state of the finite automaton passed to the parent node. The parent node can then use the information encoded in the states corresponding to its children when pre-computing its intersection with $\mathrm{cl}_M(X_2)$. We formalize this approach using the following definition.

Definition 43. *Let v be a node of an amalgam decomposition $\mathcal{T}$ of M and X be a subset of $E(M)$. A map f_v^X from $2^{J(v)} \to 2^{J(v)}$ satisfying*

$$f_v^X(Y) = \mathrm{cl}_{M(v)}\big(\big(X \cap E(M(v))\big) \cup Y\big) \cap J(v)$$

is called the type of a node v with respect to X.

When processing a node v, we can assume we are given the types f_1^X and f_2^X of the children of v and we want to determine the type of v, which we denote by $f_{v_1}^X +_{K(v)} f_{v_2}^X$. This type is then encoded into the state of the finite automaton (along with the information for which choices of $Y \subseteq J(v)$ the formula ψ holds) and is passed to the parent node. The information is then reused to determine the type of the parent node, etc. This process is captured by the following definition.

Definition 44. *Let v be a node of an amalgam decomposition $\mathcal{T}$ of a matroid M, v_1 and v_2 the children of v, and X a subset of $E(M)$. If $f_{v_1}^X$ is the type of v_1 with respect to X and $f_{v_2}^X$ is a type of v_2 with respect to X, we say that the type*

$f^X_{v_1} +_{K(v)} f^X_{v_2}$ *of* v *is* the join of $f^X_{v_1}$ and $f^X_{v_2}$ *if for every subset* Y *of* $J(v)$ *it holds that* $f^X_{v_1} +_{K(v)} f^X_{v_2} = Z \cap J(v)$*, where* Z *is the smallest subset of* $E(K(v))$ *such that*

(1) $f^X_{v_1}(Z \cap J_1(v)) = Z \cap J_1(v)$,

(2) $f^X_{v_2}(Z \cap J_2(v)) = Z \cap J_2(v)$,

(3) $Z \supseteq Y \cup (X \cap E(K(v)))$.

Lemma 35 implies that $f^X_{v_1} +_{K(v)} f^X_{v_2}$ is the type of the node v with respect to X. Observe that the type $f^X_1 +_{K(v)} f^X_2$ in the above definition is determined by $f^X_{v_1}, f^X_{v_2}, K(v)$ and $X \cap E(K(v))$ – each of which has bounded size. This implies that the computation of the type $f^X_1 +_{K(v)} f^X_2$ can be wired in the transition function of the automaton. Deciding if $Q(x_1) \in \mathrm{cl}(X_2) \cap J(v)$ is then reduced to verifying if $Q(x_1) \in f^{X_2}_v(Y)$ for a particular choice of Y.

The case where the formula ψ is of the form $\exists x : \psi_1$ is solved by a standard argument of taking the finite tree automaton recognizing ψ_1 and transforming it to a non-deterministic finite tree automaton that tries to guess the value of x (in our case, the automaton also checks if this guessed value of x lies in the set $D(v)$ of deleted elements). This automaton can in turn be simulated using a *deterministic* finite tree automaton with up to an exponential blow-up of the number of states. Note that this blow-up does not negatively affect the asymptotic running time of the algorithm when seen from the perspective of Theorem 7. This is because the formula ψ is fixed and the matroid M (along with its decomposition) is the only variable part of the input. It is true, however, that each occurrence of a universal or existential quantifier in the MSO formula ψ leads to an algorithm with a (significantly) larger multiplicative constant.

The case $\exists X : \psi$ is solved analogously.

Since the algorithm simulating the automaton on $\mathcal{T}$ spends $\mathcal{O}(1)$ time in each of the nodes of $\mathcal{T}$, there exists a linear time algorithm solving the problem from the statement of the theorem. □

5.4 Conclusion

Strozecki [81] introduced a similar parameter which uses the operation of a matroid 2-sum instead of the generalized parallel connection. However, its applicability is limited since it allows to join matroids only using separations of size at most 2 and thus any decomposition of a 3-connected matroid M has a width of $|E(M)|$. The next proposition implies that amalgamation is able to express the 2-sum operation as a special case. Therefore, the amalgam-width is a more general parameter than the one from [81].

Proposition 45. *A 2-sum of matroids M_1 and M_2 can be replaced by finitely many operations of generalized parallel connections and deletions.*

Furthermore, the amalgam-width is a generalization of the branch-width parameter for finitely representable matroids in the sense that a bound on the value of branch-width implies a bound on the amalgam-width:

Proposition 46. *If M is a matroid with branch-width k and M is representable over a finite field $\mathbf{F}$, then the amalgam-width of M is at most $|\mathbf{F}|^{3k/2}$.*

Proof. Suppose we are given a (non-trivial) branch decomposition $\mathcal{B}$ of the matroid M of width k, along with the representation of M over $\mathbf{F}$. The elements of M are vectors from $\mathbf{F}^d$ for some dimension $d \in \mathbf{N}$. Since branch decomposition

is an unrooted tree, we select an arbitrary internal node as the root node. We construct an amalgam decomposition $\mathcal{T}$ of width at most $|\mathbf{F}|^{3k/2}$. The leaves of $\mathcal{T}$ are the leaves of $\mathcal{B}$ and correspond to the same elements of M. Similarly, the internal nodes of $\mathcal{T}$ are the internal nodes of $\mathcal{B}$ and the associated matroids $K(v)$ (for $v \in \mathcal{T}$) are defined as follows. Consider an internal node $v \in \mathcal{B}$ with children v_1 and v_2. We use E_1 and E_2 to denote the set of elements of M represented by the leaves in the subtree of $\mathcal{B}$ rooted at v_1 and v_2, respectively. We also let $E' := E(M) \setminus (E_1 \cup E_2)$. Finally, we set $F \subseteq E(M)$ to be the set of all elements in at least two of the sets $\mathrm{cl}(E_1), \mathrm{cl}(E_2)$ and $\mathrm{cl}(E')$. Note that $\dim(F) \leq \frac{3}{2}k$. We construct $K(v)$ by taking as its ground set all linear combinations of vectors from F. Consequently, the sets J_v^1 and J_v^2 are $E(K(v)) \cap E_1$ and $E(K(v)) \cap E_2$, respectively.

We need to check that the conditions of Theorem 33 are met. However, every flat X in a matroid containing all d-dimensional vectors over $\mathbf{F}$ is modular, since for any flat Y we have:

$$\mathrm{r}(X \cup Y) = \dim(X \cup Y) = \dim(X) + \dim(Y) - \dim(X \cap Y) = \mathrm{r}(X) + \mathrm{r}(Y) - \mathrm{r}(X \cap Y),$$

where $\dim(\cdot)$ is the dimension of a vector subspace of $\mathbf{F}^d$.

The additional elements $E(K(v)) \setminus E(M)$ included in the construction above can be subsequently removed by including them in the set $D(u)$ at an ancestor u of v, ensuring that the decomposition represents precisely the input matroid. □

On the other hand, every amalgam decomposition can be turned into a branch decomposition:

Proposition 47. *Let M be a matroid and $\mathcal{T}$ its amalgam decomposition of width*

k. Then there exists a branch decomposition $\mathcal{B}$ of M with width at most $2k+1$.

Proof. For simplicity, we assume the sets $D^{\mathcal{T}}(v)$ to be empty for all $v \in \mathcal{T}$. If $\mathcal{T}$ is a trivial amalgam decomposition containing a single node, then there clearly is a corresponding branch decomposition satisfying the statement of the proposition. Otherwise, let r be the root of $\mathcal{T}$. We partition the ground-set of $M = M^{\mathcal{T}}(r)$ into two sets: $E_1 = E(M_1^{\mathcal{T}}(r)) \cup E(K^{\mathcal{T}}(r))$ and $E_2 = \big(E(M_2^{\mathcal{T}}(r)) \cup E(K^{\mathcal{T}}(r))\big) \setminus E_1$. By induction, we can assume there are branch decompositions $\mathcal{B}_1$ and $\mathcal{B}_2$ of the matroids $M|E_1$ and $M|E_2$, respectively. The desired branch decomposition $\mathcal{B}$ is formed by a new node r' connected to an element of $\mathcal{B}_1$ and an element of $\mathcal{B}_2$. It is left to show that the branch-width of this decomposition is at most $2k+1$. Specifically, we need to show that

$$\mathrm{r}(E_1 \cup E_2) \geq \mathrm{r}(E_1) + \mathrm{r}(E_2) - 2k.$$

Since $\mathrm{cl}(E_1)$ is a modular semiflat, Definition 32 implies:

$$\begin{aligned} r(E_1 \cup E_2) = \mathrm{r}\Big(E_1 \cup \big(E_2 \cup E(K)\big)\Big) &= \mathrm{r}(E_1) + \mathrm{r}\big(E_2 \cup E(K)\big) - \mathrm{r}\big(E(K)\big) \\ &\geq \mathrm{r}(E_1) + \mathrm{r}(E_2) - 2k. \end{aligned}$$

□

This proposition and Theorem 18 immediately imply the following algorithmic result:

Theorem 48. *For each k, there is an $\mathcal{O}(n^4)$ algorithm taking a matroid representable over a finite field $\mathbf{F}$ as its input and constructing an amalgam decomposition of width at most $6k-1$ or concluding that the matroid has amalgam-width*

at least $\frac{2}{3}\log_{|\mathbf{F}|}(k+1)$.

This allows us to remove the assumption of having the amalgam decomposition as a part of the input from Theorem 7 for matroids representable over finite fields:

Theorem 49. *For each* $k \in \mathbf{N}$ *and each formula* φ *in monadic second-order logic there is an algorithm deciding whether a matroid* M *representable over a fixed finite field satisfies* φ *in polynomial time for matroids of amalgam width bounded from above by* k.

Chapter 6

Branch-depth of matroids

This chapter introduces another matroid parameter, which we refer to as branch-depth. Compared to the case of amalgam-width defined in Chapter 5, our motivation for the introduction of this parameter is more structural than algorithmic. Branch-depth is a matroid analogue of graph tree-depth, which was applied by Nešetřil and Ossona de Mendez in their research on combinatorial limits [66, 67]. The theory of combinatorial limits is a rapidly growing field of research [2, 4, 9–11, 28, 40, 63, 73]. It has been originally introduced for dense graphs [62], but has since been extended to many other structures including hypergraphs [29] and permutations [46, 47]. Since the introduction of the theory, a series of interesting applications and connections with other areas of mathematics has been obtained [3, 41, 51, 58, 59, 74]. This motivates its extension to matroids.

Let us provide an informal picture of the situation. The aim is mostly to illustrate the position of the tree-depth parameter within the theory.

There are various notions of combinatorial limits [61] one might attempt to generalize, including graphons (dense graph limits) and graphings (sparse graph limits). In [52], we argue that the notion of a limit object called graph model-

ing [66, 67] is a good candidate for this extension. One of the obstacles of introducing matroid limits is that, when treated as hypergraphs (with hyperedges corresponding to the bases), matroids are too sparse for the dense approach of graphons and too dense for the sparse approach of graphings. The benefit of starting instead with graph modelings is that the associated notion of first-order convergence can be generalized to matroids while avoiding this problem.

In some sense, graph modelings capture more information than graphons and graphings. This however implies that stronger conditions are required in order to guarantee a modeling to exist. In particular, the authors of [66, 67] employ the tree-depth parameter and limit themselves to graphs of bounded tree-depth. Therefore, in order to extend the abovementioned theory to matroids, one would need – among other things – to resolve the issue of generalizing the tree-depth to matroids. The original notion is defined as follows.

Definition 50. The tree-depth $\mathrm{td}(G)$ of a graph G *is the smallest possible depth of a rooted tree* T *with the property that* $G \subseteq T'$*, where* T' *denotes the transitive closure of* T*, i.e. the graph with vertex set* $V(T)$ *and an edge connecting each pair of vertices* u *and* v *such that* u *is an ancestor of* v *in* T*. Any such* T *is called the* optimal tree-depth decomposition of G*.*

In the present chapter, we define branch-depth – a matroid parameter that we claim represents a notion analogous to the above definition. To substantiate this, we show that these two parameters share several fundamental properties. These include minor monotonicity (Proposition 53) and a relation between the tree-depth of a given graph and the branch-depth of the corresponding graphic matroid (Proposition 55 and Proposition 56). An algorithm efficiently computing an approximate value of the parameter for an oracle-given matroid is presented

in Section 6.3. The algorithm also produces a certifying decomposition as a part of its output.

Since structural results are outside of the scope of this thesis, we refer the reader to the result [52] for such applications of the branch-depth parameter.

6.1 Definition and basic properties

The branch-depth of a matroid is equal to the optimal height of a certain kind of decomposition tree. In the definition below and in the proofs of subsequent claims, we use $\|T\|$ to denote the number of edges of a tree T.

Definition 51. *Let M be a finite matroid. A* depth decomposition of M *is a pair (T, f), where T is a rooted tree and $f : M \to V(T)$ is a mapping such that*

(i) $r(M) = \|T\|$, *and*

(ii) $r(X) \leq \|T^*(X)\|$ *for every* $X \subseteq M$,

where $T^*(X)$ is the subtree of T formed by the union of paths from the root to all the vertices in $f(X)$. The branch-depth of a matroid M, *denoted by* $\mathrm{bd}(M)$, *is the smallest depth of a rooted tree T such that (T, f) is a depth decomposition of M.*

For any matroid M there is a trivial decomposition where the tree is a path of length $\mathrm{r}(M)$ and all the elements are mapped to the leaf. The following lemma describes a structural property of depth decomposition.

Lemma 52. *Let M be a finite matroid. If (T, f) is a depth decomposition of M, then there is a depth decomposition (T, f') such that $f'(e)$ is a leaf of T for every element e of M.*

Proof. Let (T, f) be a depth decomposition of M. For every inner node v of T, let $\ell(v)$ be a leaf of T that is a descendant of v. For every $e \in M$, define f' as follows.

$$f'(e) := \begin{cases} f(e) & \text{if } f(e) \text{ is a leaf of } T, \\ \ell(f(e)) & \text{otherwise.} \end{cases}$$

Part (i) of Definition 51 is trivial. For part (ii), note that for a subset X of the elements of M, the number of edges of $T^*(X)$ does not decrease. □

As with the notion of graph tree-depth [68] the parameter matroid branch-depth is also minor monotone.

Proposition 53. *If M' is a minor of M, then* $\mathrm{bd}(M') \leq \mathrm{bd}(M)$.

Proof. It is enough to show that if M is a matroid and e is an element of M, then the branch-depth of both M/e and $M \setminus e$ is at most $\mathrm{bd}(M)$. Fix a matroid M and $e \in E(M)$. Let (T, f) be a depth decomposition of M of depth $\mathrm{bd}(M)$. By Lemma 52, we can assume that $f(e)$ is a leaf of T for every $e \in M$.

If e is a loop in M then $M_1 := M \setminus e = M/e$. It is easy to see that for every $X \subseteq M_1$ we have $\mathrm{r}_{M_1}(X) = \mathrm{r}_M(X)$. Hence, $(T, f|_{M_1})$ is a depth decomposition of M_1.

We now assume that e is not a loop. Let $M_1 := M/e$, u be the leaf $f(e)$, and v the parent of u. Set $T_1 = T \setminus u$ and define $f_1 : M_1 \to V(T_1)$ as follows:

$$f_1(x) = \begin{cases} v & \text{if } f(x) = u, \text{ and} \\ f(x) & \text{otherwise.} \end{cases}$$

We now show that (T_1, f_1) is a depth decomposition of M_1. First of all, since e is not a loop, we have $\mathrm{r}(M_1) = \mathrm{r}(M) - 1$. Thus, $\|T_1\| = \mathrm{r}(M_1)$. Now, consider a

subset $X \subseteq M_1$. Recall that $\mathrm{r}_{M_1}(X) = \mathrm{r}_M(X \cup \{e\}) - 1$. If $u \in f_1(X)$, we employ the bound on the rank function provided by the depth decomposition of M:

$$\|T_1^*(f_1(X))\| = \|T^*(f(X \cup \{e\}))\| - 1 \geq \mathrm{r}_M(X \cup \{e\}) - 1 = \mathrm{r}_{M_1}(X).$$

Otherwise, we have

$$\|T_1^*(f_1(X))\| = \|T^*(f(X))\| \geq \mathrm{r}_M(X) \geq \mathrm{r}_{M_1}(X).$$

Let $M_2 = M \setminus e$. If e is a bridge then $M \setminus e = M/e$. Hence, we may assume that e is not a bridge in M. In this case, we claim that $(T, f|_{M_2})$ is a depth decomposition of M_2. Since the rank of M_2 equals the rank of M, we have $\mathrm{r}(M_2) = \|T\|$ and $\|T^*(f(X))\| \geq \mathrm{r}_M(X) = \mathrm{r}_{M_2}(X)$ for every $X \subseteq M_2$. □

The next proposition relates the length of circuits in the matroid to its branch-depth. This is analogous to how the graph tree-depth is related to the existence of long paths.

Proposition 54. *Let M be a matroid and g the size of its largest circuit. Then* $\mathrm{bd}(M) \geq \log_2(g)$.

Proof. By Proposition 53, it suffices to show that the branch-depth of C_d is at least $\log_2 d$, where C_d is the matroid that consists of exactly one circuit of size d. We prove this statement by induction on d.

Let (T, f) be a depth decomposition of C_d such that T has depth $\mathrm{bd}(C_d)$ and such that $f(e)$ is a leaf of T for every $e \in C_d$. Its existence follows from Lemma 52.

We first prove that the root r of T has degree 1. Suppose not. Let (T_1, T_2) be

a partition of T into two edge-disjoint subtrees both rooted at r. Then $\|T_i\| \geq \mathrm{r}(f^{-1}(V(T_i))) = |f^{-1}(V(T_i))|$ for $i \in \{1,2\}$. Hence $\mathrm{r}(C_d) = \|T\| = \|T_1\| + \|T_2\| = |C_d| = \mathrm{r}(C_d) + 1$, a contradiction.

Let v be a vertex of T of degree larger than 2 that is as close to the root r as possible. If there is no such vertex, T is a path and it has depth $d - 1 \geq \log_2(d)$.

Let P be a path from r to v, ℓ its length, and (T_1, T_2) a split of $T \setminus (P \setminus v)$ into two edge-disjoint trees rooted at v. Let $m_i = \|T_i\|$ and $n_i = |f^{-1}(V(T_i))|$ for $i \in \{1,2\}$. Observe that both n_1 and n_2 are non-zero, and that

$$m_1 + m_2 + \ell = \mathrm{r}(C_d) = d - 1 \qquad \text{and} \qquad n_1 + n_2 = |C_d| = d. \tag{6.1}$$

Since any proper subset of C_d is independent, $n_i \leq m_i + \ell$ for $i \in \{1,2\}$. By symmetry, we may assume that $n_1 \leq n_2$, which gives $n_1 \leq \frac{d}{2}$.

Let $M' := C_d / f^{-1}(V(T_1))$. Note that M' is isomorphic to C_{n_2}. Also, observe that the tree T_2 with $f|_{M'}$ is a depth decomposition of M'. By induction, the depth of T_2 is at least $\log_2 \frac{d}{2}$. We conclude that the depth of T is at least $\log_2 \frac{d}{2} + \ell = \log_2 d - 1 + \ell \geq \log_2 d$. □

We next relate the branch-depth of a graphic matroid to the tree-depth of the underlying graph.

Proposition 55. *The branch-depth of a graphic matroid $M(G)$ is at most $\mathrm{td}(G)$.*

Proof. Let G be a graph on n vertices and let $M := M(G)$ be the corresponding graphic matroid. We can assume that G is 2-connected, since otherwise we can construct the depth decompositions of the 2-connected components of G and then join them by identifying their roots.

Let T be the optimal tree-depth decomposition of G. We construct a depth decomposition (T_M, f_M) as follows. We set $T_M := T$. Next, we define the function f_M. Consider $e = \{u, v\} \in E(M)$. By symmetry, we can assume that u is an ancestor of v in T. We set the function f_M to map the matroid element e to v.

We verify the two conditions of Definition 51. Since G is connected, we indeed have $\mathrm{r}(M) = n - 1 = |V(T)| - 1$. Consider any subset $X \subseteq E(M)$ and use U to denote the set of those vertices of G that are incident to an edge in X. It suffices to show $\mathrm{r}_M(X) \leq |T^*_M(X)|$ for acyclic X, since removing an edge of a cycle in X does not decrease $\mathrm{r}_M(X)$ and also does not increase $|T^*_M(X)|$. Therefore, we have $\mathrm{r}_M(X) = |X| - c + 1$, where c is the number of components of the graph $(V(G), X)$, and $|T^*_M(X)| \geq |X|$.

□

Note that the converse of Proposition 55 does not need to hold since the graphic matroids of a star and a path with the same number of edges are identical; at the same time the graphic matroid of any star has a branch-depth of 1 and the branch-depth of the graphic matroid of a path is equal to the length of the path. Nevertheless, the following inequality holds for 2-connected graphs.

Proposition 56. *Let G be a 2-connected graph with tree-depth d. Then, the branch-depth of a graphic matroid $M(G)$ is at least $\frac{1}{2}\log_2 d$.*

Proof. Since $\mathrm{td}(G) = d$, the graph G contains a cycle of length $\sqrt{d}$, by [68, Proposition 6.2]. Therefore, by Proposition 54, $\mathrm{bd}(M(G)) \geq \frac{1}{2}\log_2 d$. □

6.2 Technical lemmas

After introducing a combinatorial parameter, it is natural to turn to the question of whether its value can be efficiently calculated or at least approximated. We give such an algorithm in Section 6.3. This section lists proofs of several technical statements that allow us to present the algorithm and its proof. Roughly speaking, the algorithm proceeds by identifying a circuit of the matroid and gradually contracting its elements. Essentially, two things can happen: either the matroid "falls apart" into several components or it remains connected after the entire circuit is contracted. In the former case, the algorithm calls itself recursively on the individual components. This case results in a branching of the decomposition tree. In the latter case, it selects another circuit and continues. That leads to an increase of the height of the decomposition tree.

In the present section, we prove lemmas that allow us to track how are circuits of a matroid affected by element contractions. Ultimately, this allows us to construct an obstruction to small branch-depth of M, which is then used to prove that the approximation ratio of the algorithm of Section 6.3.

The following claim follows directly from the definition of contraction of an element in a matroid.

Lemma 57. *Let C be a circuit in a matroid M. Let $e \in C$. If $|C| \geq 1$, then the set $C \setminus e$ is a circuit in M/e.*

When encountering a circuit, the algorithm is going to proceed by contracting one of its elements. The following lemma will be crucial for the analysis.

Lemma 58. *Let M be a connected matroid, e an element of M such that M/e is disconnected, and let $M_1, \ldots, M_k$ be components of M/e. For every circuit C*

of M containing e, there exists $i \in \{1, \ldots, k\}$ such that $C \subseteq M_i \cup \{e\}$.

Proof. Assume the contrary. Let $M_1 \cup M_2$ be a non-trivial partition of the elements of M/e such that $\mathrm{r}(M_1) + \mathrm{r}(M_2) = \mathrm{r}(M/e)$. Let $D_i := C \cap M_i \neq \emptyset$ for $i \in \{1, 2\}$. Then we have a contradiction:

$$\begin{aligned} |D_1| + |D_2| &= |C| - 1 = \mathrm{r}_M(C) = \mathrm{r}_{M/e}(C \setminus e) + 1 \\ &= \mathrm{r}_{M/e}(D_1) + \mathrm{r}_{M/e}(D_2) + 1 = |D_1| + |D_2| + 1. \end{aligned}$$

□

Lemma 57 and Lemma 58 yield the following.

Lemma 59. *Let M be a connected matroid. Let e be an element of M such that M/e is not connected and let $M_1, \ldots, M_k$ be the components of M/e. For each $i = 1, \ldots, k$ there is a circuit C_i in M containing e such that $C_i \subseteq M_i \cup \{e\}$.*

The following lemma allows us to find an obstruction to small branch-depth. We utilize them to show that Algorithm 1 always returns a depth decomposition of depth at most $4^{\mathrm{bd}(M)}$.

Lemma 60. *Let M be a matroid. Let $e_1, \ldots, e_k$ be distinct elements of M and $C_0, C_1, \ldots, C_k$ subsets of M such that*

$$\begin{aligned} |C_i| \geq 3 \quad &\text{for } i = 0, \ldots, k, \\ C_{i-1} \cap C_i = \{e_i\} \quad &\text{for } i = 1, \ldots, k, \\ C_i \cap C_j = \emptyset \quad &\text{for } |i - j| \geq 2. \end{aligned}$$

Let $e_0 \in C_0 \setminus \{e_1\}$ *and* $e'_i \in C_{i-1} \setminus \{e_{i-1}, e_i\}$, $i = 1, \ldots, k$. *Further, set*

$$M_i := \begin{cases} M & \text{for } i = 0, \\ M_{i-1}/(C_{i-1} \setminus \{e_i, e'_i\}) & \text{for } i = 1, \ldots, k. \end{cases}$$

If C_i *is a circuit in* M_i *for* $i = 0, 1, \ldots, k$, *then* M *contains a circuit of length at least* $k + 3$ *containing* e_0.

Proof. We prove the statement by induction on k. For $k = 0$ it suffices to take the circuit C_0 itself.

Let $k \geq 1$. By induction, $M_1 = M_0/(C_0 \setminus \{e_1, e'_1\})$ contains a circuit C of length at least $k + 2$ that contains e_1. Let $D = C \setminus e_1$. Since C is a circuit, D is independent in M_1 and thus in M_0. Also note $|D| \geq k + 1$.

Let $N = M_0/(C_0 \setminus \{e_0, e_1, e'_1\})$. Since C_0 is a circuit in M_0, $\{e_0, e_1, e'_1\}$ is a circuit in N. Furthermore, it holds that $M_1 = N/e_0$. If Y is a circuit in N, then there is a circuit $Y' \supseteq Y$ in M. Therefore, it suffices to find a circuit of length at least $k + 3$ in N. We will show that $D \cup \{e_0, e'_1\}$ or $D \cup \{e_0, e_1\}$ is a circuit in N.

Since D is independent in N/e_0, we get that $D \cup \{e_0\}$ is independent in M. We have $\mathrm{r}_N(X \cup \{e_i, e_j\}) = \mathrm{r}_N(X \cup \{e_0, e_1, e'_1\})$ for any $e_i, e_j \in \{e_0, e_1, e'_1\}$, $e_i \neq e_j$ and for any set $X \subseteq N$ by the submodularity of the rank function:

$$\mathrm{r}_N(\{e_i, e_j\}) + \mathrm{r}_N(X \cup \{e_0, e_1, e'_1\}) \leq \mathrm{r}_N(X \cup \{e_i, e_j\}) + \mathrm{r}_N(\{e_0, e_1, e'_1\}).$$

Hence, for any proper subset $D' \subsetneq D$ we have

$$\mathrm{r}_N(D' \cup \{e_0, e'_1\}) = \mathrm{r}_N(D' \cup \{e_0, e_1\}) = \mathrm{r}_{M_1}(D' \cup \{e_1\}) + 1 = |D'| + 2,$$

where the last equality follows from the fact that $D \cup \{e_1\}$ is a circuit in M_1. Thus, both $D' \cup \{e_0, e_1'\}$ and $D' \cup \{e_0, e_1\}$ are independent in N. On the other hand, it also holds that

$$\mathrm{r}_N(D \cup \{e_0, e_1'\}) = \mathrm{r}_N(D \cup \{e_0, e_1\}) = \mathrm{r}_{M_1}(D \cup \{e_1\}) + 1 = |D| + 1,$$

since $D \cup \{e_1\}$ is a circuit in M_{M_1}. Consequently, neither $D \cup \{e_0, e_1\}$ nor $D \cup \{e_0, e_1'\}$ is independent. It suffices to prove that $D \cup \{e_1\}$ or $D \cup \{e_1'\}$ is independent in N. Indeed,

$$\mathrm{r}_N(D \cup \{e_1\}) + \mathrm{r}_N(D \cup \{e_1'\}) \geq \mathrm{r}_N(D) + \mathrm{r}_N(D \cup \{e_1, e_1'\}) = 2|D| + 1.$$

The proof is now complete. □

We get the following corollary.

Corollary 61. *Let M be a matroid. If $C_0, C_1, \ldots, C_k$ and $M_0, \ldots, M_k$ are as in Lemma 60, then the matroid M contains a circuit of length at least $\sqrt{\sum_{i=0}^{k} |C_i|}$.*

Proof. Let $t := \sum_{i=0}^{k} |C_i|$. If $t \leq (k+1)^2$, then by Lemma 60 there is a circuit of length at least $k + 3 > \sqrt{t}$. On the other hand, if $t > (k+1)^2$, then there exists $i \in \{0, 1, \ldots, k\}$ such that $|C_i| \geq \frac{t}{k+1} > \sqrt{t}$. □

6.3 Approximating branch-depth

We now present our polynomial time algorithm for constructing a depth decomposition of an oracle-given matroid M with depth at most $4^{\mathrm{bd}(M)}$. The pseudocode is given as Algorithm 1 in the form of a routine taking three parameters:

a connected matroid M, one of its circuits C, and a non-loop element $e \in C$. For disconnected matroids we process the components individually and glue the resulting depth decompositions by identifying their roots. Note that every connected matroid has a circuit and a non-loop element in it unless $|M| = 1$.

If the rank of M is at most one, the routine returns the trivial depth decomposition. Assume $\mathrm{r}(M) \geq 2$. If $|C| \leq 2$, we find another circuit containing e of size at least three (the existence of such circuit in a connected matroid with rank at least two is implied by the definition of connectivity). We assume $|C| \geq 3$, proceed by contracting e in M and analysing the resulting matroid. If it is connected, the algorithm calls itself recursively (Step 3 of Algorithm 1) on the contracted matroid. The recursive call is made for the contracted matroid, its circuit $C \setminus \{e\}$, and a non-loop element e_1 of the circuit. The existence of these is guaranteed by the assumption $|C| \geq 3$. After the call is finished, we alter the resulting decomposition by adding a new root.

If M/e is not connected, the recursive calls are performed on each component separately (Step 4 and Step 5). The resulting decomposition is obtained by identifying the roots of the individual decompositions.

It is easily verified that the algorithm finishes in time polynomial in the number of elements of the input matroid: if the recursive call in Step 2 is executed, the next execution avoids this step and instead either immediately returns a trivial decomposition or performs one of the other recursive calls, which in turn lead to a decrease in the input size. If Step 3 is reached, only a single recursive call is made by the routine, in which the number of matroid elements is decreased by one. If Steps 4 and 5 are reached the number of recursive calls equals the number of connected components with the sizes of the corresponding inputs distributed

proportionally to the component size.

We next establish that the obtained depth decomposition is valid.

Lemma 62. *Algorithm 1 returns a valid depth decomposition of M.*

Proof. Let M be the input matroid and (T, f) the output of the Algorithm 1. Clearly, T is a tree and f a mapping from $E(M)$ to $V(T)$. Thus, we need to verify the two conditions from Definition 51. Establishing the first condition, $\mathrm{r}(M) = ||T||$, is straightforward: we contract a non-loop element of the matroid precisely when we are decreasing the rank of the currently processed matroid. Indeed, whenever Step 2 of the routine is reached, the recursive call is performed on the same input and the resulting output is returned without extending the decomposition. Whenever Step 3 is reached, the matroid rank is decreased by one for the recursive call and the output decomposition is extended by a single edge. When Steps 4 and 5 are reached, the contracted matroid is split into connected components, and the resulting depth decompositions are glued together with a single edge introduced. Note that the contraction of the edge e results in the sum of ranks of the individual components being equal to $\mathrm{r}(M) - 1$. Therefore, we get $\mathrm{r}(M) = ||T||$.

To establish the second condition from Definition 51, we proceed by induction on the size of M. Assume X is a subset of $E(M)$. In the trivial case when M has rank at most 1, the inequality $\mathrm{r}(X) \leq ||T^*(X)||$ is clearly satisfied. Suppose therefore that Step 3 is reached, i.e., the matroid M/e is connected. From the induction hypothesis we get $\mathrm{r}(X \setminus \{e\}) \leq {T_0}^*(X \setminus \{e\})$, where T_0 is the depth decomposition of M/e returned by the recursive call. Since T is obtained by

Algorithm 1: construct(M, C, e)
Input: a connected matroid M, a circuit C of M, and a non-loop element $e \in C$
Output: a depth decomposition of M

```
if r(M) = 0 then
Step 0  |  return one-vertex tree with f mapping all elements to the root;
else
    if r(M) = 1 then
Step 1  |  return one-edge tree with f mapping all elements to the leaf;
    else
        if |C| = 2 then
Step 2  |  choose C' satisfying e ∈ C' and |C'| ≥ 3;
        |  return construct(M, C', e);
        else
            if M/e is connected then
Step 3  |  choose a non-loop element e_1 ∈ C \ e;
        |  (T', f') := construct(M/e, C \ e, e_1);
        |  T := (v(T') ∪ {v_0}, E(T') ∪ {v_0 r}) where r is the root of T',
        |  rooted at v_0;
        |  f(e) := r; f(e') := f'(e') for e' ≠ e;
        |  return (T, f);
            else
Step 4  |  for the component M_0 of M/e containing C \ e do
        |  |  choose a non-loop element e_0 ∈ C \ e;
        |  |  (T_0, f_0) := construct(M_0, C \ e, e_0);
        |  end
Step 5  |  for each component M_i of M/e disjoint from C do
        |  |  choose a circuit C_i of M contained in M_i ∪ {e} that
        |  |  contains e;
        |  |  choose e_i ∈ C_i \ e;
        |  |  (T_i, f_i) := construct(M_i, C_i \ e, e_i);
        |  end
        |  identify all the roots r_i of T_i into a single r, obtaining T';
        |  T := (v(T') ∪ {v_0}, E(T') ∪ {v_0 r}), choosing v_0 as the root;
        |  f(e) := r, f(e_i) := f_i(e_i) for e_i ∈ M_i;
        |  return (T, f);
            end
        end
    end
end
```

adding a new root and connecting it by an edge with the old root, we get:

$$\mathrm{r}(X) \leq \mathrm{r}(X \setminus \{e\}) + 1 \leq \|T_0^*(X \setminus \{e\})\| + 1 = \|T^*(X)\| .$$

Finally, suppose that Steps 4 and 5 are executed, i.e., the matroid M/e is divided into components $M_1, \ldots, M_k$. By induction, we get $\mathrm{r}_{M_i}(X \cap E(M_i)) \leq \|T_i^*(X \cap E(M_i))\|$, where T_i is the depth decomposition of M_i returned by the recursive call for M_i. Since the resulting depth decomposition is constructed by identifying the roots of $T_1, \ldots, T_k$ and connecting them to a new root node, we get

$$\mathrm{r}(X) \leq 1 + \sum_i \mathrm{r}_{M_i}(X \cap E(M_i)) \leq 1 + \sum_i \|T^*(X \cap E(M_i))\| = \|T^*(X)\| .$$

□

Lemma 63. *Algorithm 1 returns a depth decomposition of M with depth at most $4^{\mathrm{bd}(M)}$.*

Proof. Let d be the depth of the depth decomposition T returned by the algorithm for a matroid M. Let $r = v_0, v_1, \ldots, v_d$ be a path in T of length d from the root to one of the leaves. It is easy to see that each vertex of T which is not a leaf is the root of some subtree of T during the execution of the algorithm. For $i = 0, 1, \ldots, d-1$ let (M_i, C_i, e_i) be the matroid (together with a circuit and an element of it) such that its decomposition tree is the subtree of T rooted at v_i with the root of degree one which contains $v_i, v_{i+1}, \ldots, v_d$ during the execution of the algorithm. Clearly, $M_0 = M$ and $C_0 = C$. Moreover, $\mathrm{r}(M_{d-1}) = 1$ and $|C_{d-1}| = 2$.

For every $i = 0, \ldots, d-3$ precisely one of the following cases occurs:

- $M_{i+1} = M_i/e_i$, $C_{i+1} = C_i \setminus e_i$ (Step 3),
- M_{i+1} is a component of M_i/e_i, $C_{i+1} = C_i \setminus e_i$ (Step 4), or
- M_{i+1} is a component of M_i/e_i, $C_{i+1} \cap C_i = \emptyset$, but $C_{i+1} \cup \{e_i\}$ is a circuit in M_i (Steps 2 or 5 of the algorithm).

Note that in all the cases we have $f(e_i) = v_{i+1}$.

Let $C_{j_0}, \ldots, C_{j_k}$ be the subsequence of inclusion-wise maximal sets among $C_0, \ldots, C_{d-1}$. These are pairwise disjoint due to the fact that the algorithm proceeds by contracting elements (possibly with an additional branching into a particular component).

Let $C'_0 := C_{i_0} = C_0$ and $e'_0 := e_0$; let $C'_i := C_{j_i} \cup \{e_{j_i-1}\}$ and $e'_i := e_{j_i-1}$ for $i = 1, \ldots, k$. The circuits $C'_0, \ldots, C'_k$ and the elements $e'_0, \ldots, e'_k$ fulfil the conditions of Lemma 61. Since each C'_i is contracted at most $|C'_i - 2|$ times, we have $\sum_{i=0}^{k} |C'_i| > d$.

By Lemma 61, M has a circuit of length at least $\sqrt{\sum_{j=0}^{k} |C_{i_j}|} > \sqrt{d}$. By Proposition 54 we have $\mathrm{bd}(M) \geq \frac{1}{2} \log_2 d$. □

6.4 Conclusion

The value of the branch-depth parameter is closely related to the length of the largest circuit. Proposition 54 provides a lower bound while the analysis of the previous section yields the following upper bound.

Corollary 64. *The branch-depth of a finite matroid M is at most ℓ^2, where ℓ is the size of the largest circuit of M.*

For some applications, a depth decomposition with a particular property might be more suitable from a technical point of view. In Lemma 52, we have seen we can assume all the elements of the matroid in question to be mapped to the leafs of the decomposition tree. Algorithm 1 guarantees the existence of another type of depth decomposition:

Corollary 65. *For each matroid M, there is a depth decomposition (T, f) and a basis B of M such that $f|_B$ is a bijection from B to $V(T) \setminus \{r\}$, and $f(e)$ is a leaf for every $e \notin B$. Furthermore, the depth of T is at most $4^{\mathrm{bd}(M)}$.*

This is implied by the fact that each inner node of the tree returned by the algorithm corresponds to an element of a circuit of the matroid. Depth decompositions that feature additional properties such as the one above have a potential to simplify various technical arguments. Our structural proofs in [52] are an example of this.

Chapter 7

Conclusion and future work

This thesis presents results from several different corners of parameterized complexity. These include both constructive ones and negative ones:

- The non-existence of a polynomial time kernel for Permutation Pattern Matching, under a realistic complexity-theoretic assumption.
- The non-existence of a subexponential time algorithm for the Optimum Linear Arrangement problem, under the assumption of a non-existence of a subexponential time approximation scheme for Min Bisection.
- The introduction of a novel matroid parameter – amalgam-width – along with the corresponding Courcelle-like meta-algorithm.
- The introduction of another matroid parameter – branch-depth – along with an efficient algorithm approximating its value on an oracle-given matroid.

Proving complexity upper bounds by designing efficient algorithms and establishing lower bounds by designing efficient reductions is often connected. When trying to devise a suitable reduction (e.g., from 3-SAT when proving non-existence

of a subexponential algorithm under the ETH) it is helpful to look at the state-of-the-art algorithm and consider its bottleneck – the subroutine that consumes the largest portion of the runtime. One might then try to exploit this subroutine in order to solve the original problem. Correspondingly, a reduction proving a certain hardness property hints at what all algorithms will need to overcome when solving the problem.

This might be useful for resolving a natural open question concerning Permutation Pattern Matching: is there a parameterized algorithm solving the problem in $2^{\mathcal{O}(\ell)}n^{\mathcal{O}(1)}$ time? As mentioned in Chapter 3, the currently fastest known parameterized algorithm [36] for the problem takes $2^{\mathcal{O}(\ell^2)}n$ steps. Alternatively, one might want to establish an ETH lower bound for the running time of any parameterized algorithm for the problem. This would be done by constructing a reduction from 3-SAT that results in instances where the length of the pattern ℓ is bounded by a function of $v+s$, where v and s are the number of variables and clauses, respectively, of the original instance of 3-SAT. The slower this function grows the tighter lower bound is obtained. Specifically, a polynomial time reduction that results in instances with parameter $\mathcal{O}(g(v+s))$, where $g(\cdot)$ is a function on $\mathbf{N}$, implies the non-existence of an algorithm solving the problem in $2^{o(g^{-1}(\ell))} \cdot n^{\mathcal{O}(1)}$ steps, where $g^{-1}(\cdot)$ is the inverse of $g(\cdot)$.

Our proof of kernelization lower bounds for Permutation Pattern Matching employs a new polynomial time reduction that is specifically tailored for cross-composition. It also leads to a new proof of NP-hardness of Permutation Pattern Matching. It is interesting to compare our reduction with the original one from [12]. The original reduction starts from 3-SAT and also establishes #P-hardness. This is because there is a one-to-one correspondence between the

solutions to the instance of 3-SAT and the occurrences of the pattern in the resulting instance of Permutation Pattern Matching. Our reduction is quite different in this regard. Inspecting the proof of Lemma 22, we see that we "control" the position of only a portion of the occurrences of the pattern's entries within the second permutation. Specifically, consider a pair of permutations $\pi_z(K_l)$ and $\pi_z(G)$ as in the statement of Lemma 22 and assume that the former is a pattern of the latter. The proof constrains where all the encoding entries and also the middle entries of the individual separating runs get mapped by the certifying mapping. However, nothing is claimed about the position of the rest of the pattern entries within the target permutation. Indeed, given a mapping certifying the pattern $\pi_z(K_l)$ in $\pi_z(G)$ it will typically be possible to shift where at least some of the separating runs of $\pi_z(K_l)$ are mapped within $\pi_z(G)$ without invalidating any constrains of Definition 19. It can be easily verified that this is particularly true if the clique is not detected on vertices of G with indices forming a consecutive set (i.e., with indices $\{i, \ldots, i + l - 1\}$ for some $i \in \mathbf{N}$). In such cases the entries of separating runs of some vertices of K_l can be remapped to encoding entries of the skipped vertices (i.e., those not in the detected clique) of G and the result is still a valid mapping certifying the pattern. This way, a large number of additional occurrences of the pattern can be constructed in almost all cases.

Therefore, in some sense, our polynomial reduction is "robust" – a single clique in the input graph leads to many different occurrences of the pattern (for almost all inputs). This can be easily extended to *all* inputs by a straightforward generalization of our reduction. First note that instead of searching for $\pi_z(K_l)$ in $\pi_z(G)$, we can use $\pi_z(K_l)$ and $\pi_{z'}(G)$ for $z' := \lceil 1.1z \rceil$. The proofs remain identical.

Secondly, our reduction can be extended straightforwardly to cases where G is a multigraph (i.e., edge multiplicities are allowed). Running Permutation Pattern Matching instead on $\pi_z(K_l)$ in $\pi_{z'}(G')$, where $z' := \lceil 1.1z \rceil$ and G' is a multigraph obtained from G by giving each edge, e.g., constant multiplicity, then ensures that the aforementioned robustness property can be guaranteed. A random deletion of an element from the target permutation then does not affect the outcome.

A common property of the two reductions is that they both use long decreasing subsequences within both of the permutations of the resulting instance as separators. These ensure that certain sets of entries of the pattern permutation are mapped in the same part of the target permutation. For example, the original reduction from [12] considers a 3-CNF formula and creates a pair of permutations, both of which have two parts: the first encodes the variables of the formula and the second corresponds to its clauses. A separator is inserted between these parts to enforce that all entries of the first part of the first permutation map to the first part of the other permutation (and analogously for the second part). However, the two reductions differ significantly in how these separators are constructed. In [12], the reduction places a long monotonic subsequence of unique length in both of the resulting permutations. Since neither of them contains any other monotonic subsequence of such length, these two subsequences must be mapped to each other by the certifying function (provided one actually exists). The individual elements of this long subsequence are then used as separators. This approach cannot be employed in our case since the number of our separators grows with the size of the graph G (which is the input graph of the Clique problem) and we want to avoid a dependence of the pattern length

(i.e., the length of the first permutation) on the size of G. Indeed, the length of the pattern in our reduction depends only on the size of the maximum connected subgraph of G. We hope this provides further insight into the complexity of the Permutation Pattern Matching problem.

Our second non-existence result is conditioned on Conjecture 3, which states that there is no subexponential time approximation scheme for Min Bisection on d-regular graphs, for some fixed $d \in \mathbf{N}$. This conjecture could possibly be refuted by the construction of such approximation scheme. On the other hand, it would be interesting to reduce this conjecture to one of the more established complexity hypotheses, e.g., the ETH.

We have attempted to connect Conjecture 3 to a hypothesis of Feige [32], which is concerned with the hardness of distinguishing satisfiable instances of 3-SAT from the "typical" instances of this problem:

Hypothesis 66 (Feige, [32])**.** *Even when Δ is an arbitrarily large constant independent of v, there is no polynomial time algorithm that rejects most instances of 3-*SAT *with v variables and $\Delta \cdot v$ clauses, and never wrongly rejects a satisfiable instance of 3-*SAT*.*

The following variant on this conjecture is also introduced there:

Hypothesis 67 (Feige, [32])**.** *For every fixed $\varepsilon > 0$, for Δ sufficiently large constant independent of v, there is no polynomial time algorithm that rejects most instances of 3-*SAT *with v variables and $s = \Delta \cdot v$ clauses, but never rejects an instance of 3-*SAT *with $(1 - \varepsilon) \cdot s$ satisfiable clauses.*

Feige provides some rationale in favor of these hypotheses and also some reasoning against them. Still, the validity of both remains open. The latter

hypothesis is weaker. The concept of R-3-SAT-hardness is then introduced:

Definition 68. *A computational problem is* R-*3*-SAT-*hard if a polynomial time algorithm for the problem contradicts Hypothesis 67.*

It is then deduced that the problem of approximating Min Bisection within a ratio below $\frac{4}{3}$ is R-3-SAT-hard. This is significant because of the present lack of other approximation-hardness results for this problem. A similar hardness result for the problem of finding the densest subgraph on a specified number of vertices in a given graph is also obtained in [32].

We have found no immediate reason violating Hypothesis 67 when strengthened to subexponential time. Perhaps, it could be possible to deduce Conjecture 3 from such a variation of Feige's hypothesis. Unfortunately, we have encountered a series of obstacles when trying to pursue this direction. For example, one would need to enforce the regularity of the graphs in the resulting instance of Min Bisection and also reduce the quadratic blowup of the instance size exhibited by Feige's reduction.

In Chapter 5 and Chapter 6 we introduce two parameters for matroids. The definitions of the parameters are motivated by recent research in graph theory. Extending graphic notions and theorems to matroids is a major theme in matroid theory that arguably lies behind some of the most interesting results in the area. Probably the best recent example of this would be the announced proof of Rota's conjecture – a major matroid conjecture – by Geelen, Gerards, and Whittle [38]. The proof utilizes an extension of the graph minors project [75] to matroids, which was also announced by the authors of [38].

The fields of graph limits and graph decompositions are further examples of areas that currently receive a considerable degree of interest by the research

community and have some potential of providing new insights into matroids.

The goal of Chapter 5 is the introduction of a natural matroid width parameter for which it would be possible to extend the Courcelle-like algorithmic results to matroids that are not finitely-representable. In some ways, finitely-representable matroids still "behave" similarly to graphs. For example the matroid analogue of the Courcelle theorem, the theorem of Hliněný [42] (stated in this thesis as Theorem 27), requires the input matroid to be finitely-represented in addition to having a bounded branch-width. In some sense this is natural since providing a representation over a finite field is an efficient way of specifying a matroid as an algorithmic input. However, results such as the non-existence of subexponential time algorithms for detecting U_2^4 minors in general matroids [79] indicate that it is only behind the boundary of finite representability where the true generality of matroids reveals itself.

Similar motivation can be found behind the decomposition-width parameter introduced in [57]. Let us briefly sketch the definition of this parameter. The decomposition on which the parameter is based is called *a K-decomposition*. It is a rooted tree in which all inner nodes have exactly two children and each inner node is labelled by two functions, both of them defined on the set $\{0, \ldots, K\}^2$. The leaves of the tree correspond to the elements of the decomposed matroid. Information about whether the element is a loop is stored in its leaf. The purpose of the two functions is as follows. One of the functions encodes how the rank function of the matroid behaves when combining subsets of elements from the leaves descending from the left child with subsets of elements corresponding to the leaves in the right subtree. It calculates the rank of a subset of matroid elements based on a limited amount of information passed from its children and encoded

in the form of a number between 0 and K. Determining what information should the currently processed node send to its parent is the purpose of the second function. The function receives this limited information from both of its children and returns the value sent to its parent (the leaves of the tree send a number that depends chiefly on whether the corresponding element is contained in the subset whose rank is being calculated). In this way, the computation proceeds towards the root of the decomposition at which point the rank of the subset is revealed. The width of the decomposition is equal to K.

The notion of a K-decomposition is a very general abstraction of the calculations that one might be doing when computing the value of the rank function of the decomposed matroid. However, it does have the disadvantage that the way in which the matroid is decomposed into the individual pieces does not correspond to any fundamental matroid-theoretic notion. In contrast to this, the notion of generalized parallel connection employed in amalgam decompositions is very natural. The downside is that it is not guaranteed to exist for a general pair of matroids. Perhaps one might devise another matroid gluing operation for which this situation is more balanced.

As already stated, Chapter 6 is motivated by an extension of the theory of combinatorial limits to matroids. One of the goals of this area is to design combinatorial limit objects – structures (typically uncountably infinite ones) that encode extremal properties of discrete structures such as graphs. This can be useful when proving, e.g., an extremal statement about some class of graphs: instead of dealing with the set of all graphs from the class, one can consider the set of limit objects (e.g., a set of graphings or a set of graphons [61]) encoding the extremal properties of these graphs. A proof of a corresponding statement for all

such limit objects then implies the proof of the original extremal statement for all graphs. Results of this type typically stem from the particularly developed area of dense limits (examples include [3, 58, 59, 74]).

The branch-width parameter has been used in [52] to establish a theorem guaranteeing the existence of a limit object called matroid modeling (specifically a variant of an infinite matroid based on the result [13]) under conditions analogous to the result of Nešetřil and Ossona de Mendez [67]. The statement of the theorem is as follows:

Theorem 69 (Král' et al., [52])**.** *Every first-order convergent sequence of matroids with bounded branch-depth that is representable over a fixed finite field has a limit matroid modeling.*

This can be compared to the theorem that motivated this work:

Theorem 70 (Nešetřil and Ossona de Mendez, [67])**.** *Every first-order convergent sequence of graphs with bounded tree-depth has a limit modeling.*

(In the statement of the respective theorems, first-order convergence corresponds to the following condition: for each first-order formula, the probability that a uniformly random assignment of graph vertices or matroid elements to the free variables satisfies the formula converges.) Clearly, the first result is a direct analogy of the second one.

Independently of us, Devos and Oum (private communication) investigated a similar matroid parameter inspired by the work of Ditmann and Oporowski [26]. Their definition is recursive: a matroid formed only by loops and co-loops is assigned the parameter value 0, a general matroid is assigned the minimum value k such that there is an edge that can be contracted to obtain a matroid with

parameter value $k-1$. Like branch-depth, such a parameter is also related to the length of the largest circuit. This parameter has applications in combinatorial optimization.

List of Abbreviations

CNF	...	Conjunctive Normal Form
CSP	...	Constraint Satisfaction Problem
ETH	...	Exponential Time Hypothesis
FPT	...	Fixed-Parameter Tractable
MSO	...	Monadic Second-Order
OLA	...	Optimum Linear Arrangement
SAT	...	Satisfiability problem

Bibliography

[1] S. Ahal, Y. Rabinovich. *On Complexity of the Subpattern Problem.* SIAM J. Discrete Math. 22, 2008, 629–649.

[2] D. Aldous, R. Lyons. *Processes on unimodular random networks.* Electron. J. Probab. 12, 2007, 1454–1508.

[3] R. Baber, J. Talbot. *Hypergraphs do Jump.* Comb. Probab. and Comput. 20, 2011, 161–171

[4] I. Benjamni, O. Schramm. *Recurrence of distributional limits of finite planar graphs.* Electron. J. Probab. 6, 2001, 1–13.

[5] I. Bliznets, M. Cygan, P. Komosa, L. Mach. *Kernelization lower bound for Permutation Pattern Matching.* Inform. Process. Lett. 115, 2015, 527–531.

[6] I. Bliznets, M. Cygan, P. Komosa, L. Mach, M. Pilipczuk. *Lower bounds for the parameterized complexity of Minimum Fill-in and other completion problems.* To be presented at SODA 2016, preprint available as CoRR, arXiv:1508.05282, 2015.

[7] H. L. Bodlaender, R. G. Downey, M. R. Fellows, D. Hermelin. *On problems without polynomial kernels.* J. Comput. Syst. Sci. 75, 2009, 423–434.

[8] H. L. Bodlaender, B. M. P. Jansen, S. Kratsch. *Cross-Composition: A New Technique for Kernelization Lower Bounds.* Proc. STACS, 2011, 165–176.

[9] C. Borgs, J.T. Chayes, L. Lovász, V.T. Sós, K. Vesztergombi. *Convergent sequences of dense graphs I: Subgraph frequencies, metric properties and testing.* Adv. Math. 219, 2008, 1801–1851.

[10] C. Borgs, J.T. Chayes, L. Lovász, V.T. Sós, K. Vesztergombi. *Convergent sequences of dense graphs II. Multiway cuts and statistical physics.* Ann. of Math. 176, 2012, 151–219.

[11] C. Borgs, J. Chayes, L. Lovász, V.T. Sós, B. Szegedy, K. Vesztergombi. *Graph limits and parameter testing.* Proc. STOC, 2006, 261–270.

[12] P. Bose, J. F. Buss, A. Lubiw. *Pattern Matching for Permutations.* Inform. Process. Lett. 65, 1998, 277–283.

[13] H. Bruhn, R. Diestel, M. Kriesell, R. Pendavingh, P. Wollan. *Axioms for infinite matroids.* Adv. Math. 239, 2013, 18–46.

[14] M.-L. Bruner, M. Lackner. *A fast algorithm for permutation pattern matching based on alternating runs.* SWAT 2012, Lecture Notes in Computer Science 7357, Springer, 2012, 261–270.

[15] M.-L. Bruner, M. Lackner. *The computational landscape of permutation patterns.* Pure Mathematics and Applications 24, 2013, 83–101.

[16] T. N. Bui, C. Heigham, C. Jones, F. T. Leighton. *Improving the performance of the Kernighan-Lin and simulated annealing graph bisection algorithms.* Proc. DAC, 1989, 775–778.

[17] T. N. Bui, L. C. Strite. *An ant system algorithm for graph bisection.* Proc. GECCO, 2002, 43–51.

[18] T. N. Bui, S. Chaudhuri, F. T. Leighton, and M. Sipser. *Graph bisection algorithms with good average case behavior.* Combinatorica 7, 1987, 171–191.

[19] B.-M. Bui-Xuan, J. A. Telle, M. Vatshelle. *Boolean-width of graphs.* Proc. IWPEC, 2009, Lecture Notes in Compute r Science 5917, Springer, 2009, 61–74.

[20] J.-Y. Cai, V. Chakaravarthy, L. Hemaspaandra, M. Ogihara. *Some Karp-Lipton-type theorems based on S2.* Univ. of Rochester Computer Science Technical Report TR-759, 2001.

[21] B. Courcelle. *The monadic second-order logic of graph I. Recognizable sets of finite graphs.* Inform. and Comput. 85, 1990, 12–75.

[22] B. Courcelle, J. A. Makowsky, U. Rotics. *On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic.* Discrete Appl. Math. 108, 2001, 23–52.

[23] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, S. Saurabh. *Parameterized Algorithms.* Springer, 2015.

[24] M. Cygan, D. Lokshtanov, M. Pilipczuk, M. Pilipczuk, S. Saurabh. *Minimum Bisection is Fixed Parameter Tractable.* Proc. STOC, 2014, 323–332.

[25] R. Diestel. *Graph Theory.* Springer, 2010.

[26] J. Ditmann, B. Oporowski. *Unavoidable minors of graphs of large type.* Discrete Math. 248, 2002, 27–67.

[27] R. G. Downey, M. R. Fellows. *Parameterized Complexity.* Springer, 1999.

[28] G. Elek. *On limits of finite graphs.* Combinatorica 27, 2007, 503–507.

[29] G. Elek, B. Szegedy. *A measure-theoretic approach to the theory of dense hypergraphs.* Adv. Math. 231, 2012, 1731–1772.

[30] U. Feige, R. Krauthgamer. *A polylogarithmic approximation of the minimum bisection.* SIAM J. Comput. 31, 2002, 1090–1118.

[31] U. Feige, R. Krauthgamer, K. Nissim. *Approximating the minimum bisection size (extended abstract).* Proc. STOC, 2000, 530–536.

[32] U. Feige. *Relations between average case complexity and approximation complexity.* Proc. STOC, 2002, 534–543.

[33] J. Flum, M. Grohe. *Parameterized Complexity Theory.* Springer, 2006.

[34] F. V. Fomin, Y. Villanger. *Subexponential Parameterized Algorithm for Minimum Fill-in.* Proc. SODA, 2012, 1737–1746.

[35] L. Fortnow, R. Santhanam. *Infeasibility of instance compression and succinct PCPs for NP.* J. Comput. Syst. Sci. 77, 2011, 91-106.

[36] J. Fox. *Stanley-Wilf limits are typically exponential.* To appear in Adv. Math., preprint available as CoRR, arXiv:1310.8378, 2014.

[37] T. Gavenčiak, D. Král', S. Oum. *Deciding first order logic properties of matroids.* Proc. ICALP, 2012, Lecture Notes in Computer Science 7392, Springer, 2012, 239–250.

[38] J. Geelen, B. Gerards, G. Whittle. *Solving Rota's conjecture.* Notices of the Amer. Math. Soc., 2014, 736–743.

[39] S. Guillemot, D. Marx. *Finding Small Patterns in Permutations in Linear Time.* Proc. SODA, 2014, 82–101.

[40] H. Hatami, L. Lovász, B. Szegedy. *Limits of locally-globally convergent graph sequences.* Geom. and Funct. Anal. 24, 2014, 269–296.

[41] H. Hladký, A. Mathé, V. Patel, O. Pikhurko. *Poset limits can be totally ordered.* T. Am. Math. Soc. 367, 2015, 4319-4337

[42] P. Hliněný. *Branch-width, parse trees and monadic second-order logic for matroids.* J. Comb. Theory B 96, 2006, 325–351.

[43] P. Hliněný, S. Oum. *Finding branch-decomposition and rank-decomposition.* SIAM J. Computing 38, 2008, 1012–1032.

[44] P. Hliněný, G. Whittle. *Matroid tree-width.* European J. Combin. 27, 2006, 1117–1128.

[45] S. Hoory, N. Linial, A. Widgerson. *Expander graphs and their applications.* B. Am. Math. Soc. 43, 2006, 439–561.

[46] C. Hoppen, Y. Kohayakawa, C. G. Moreira, B. Ráth, R. M. Sampaio. *Limits of permutation sequences.* J. Comb. Theory B 103, 2013, 93–113.

[47] C. Hoppen, Y. Kohayakawa, C. G. Moreira, R. M. Sampaio. *Limits of permutation sequences through permutation regularity.* Preprint available as CoRR, arXiv:1106.1663, 2011.

[48] L. Ibarra. *Finding pattern matchings for permutations.* Inform. Process. Lett. 61, 1997, 293–295.

[49] R. Impagliazzo, R. Paturi. *On the complexity of k-sat, J. Comput.* Syst. Sci. 62, 2001, 367–375.

[50] R. Impagliazzo, R. Paturi, F. Zane. *Which problems have strongly exponential complexity.* J. Comput. Syst. Sci. 63, 2001, 512–530.

[51] S. Janson. *Poset limits and exchangeable random posets.* Combinatorica 31, 2011, 529–563.

[52] F. Kardoš, K. Král', A. Liebenau, L. Mach. *First-order convergence of matroids.* Preprint available as CoRR, arXiv:1501.06518, 2015.

[53] S. Khot, N. K. Vishnoi. *The unique games conjecture, integrality gap for cut problems and embeddability of negative type metrics into L1.* Proc. FOCS, 2005, 53–62.

[54] M. Klazar. *The Füredi-Hajnal conjecture implies the Stanley-Wilf conjecture.* Formal Power Series and Algebraic Combinatorics, 2000, 250–255.

[55] D. E. Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms.* 2nd edition, Addison-Wesley, 1973.

[56] D. Král'. *Computing representations of matroids of bounded branch-width.* Proc. STACS, 2007, Lecture Notes in Computer Science 4393, Springer, 2007, 224–235.

[57] D. Král'. *Decomposition width of matroids.* Discrete Appl. Math. 160, 2012, 913–923, Lecture Notes in Computer Science 6198, Springer, 2010, 55–66.

[58] D. Král', O. Pikhurko. *Quasirandom permutations are characterized by 4-point densities.* Geom. Funct. Anal. 23, 2013, 570–579.

[59] D. Král', L. Mach, J.-S. Sereni. *A new lower bound based on Gromov's method of selecting heavily covered points.* Discrete Comput. Geom. 48, 2012, 487–498.

[60] L. Lovász. *Combinatorial Problems and Exercises.* North-Holland, 1979.

[61] L. Lovász. *Large networks and graph limits.* AMS, 2012.

[62] L. Lovász, B. Szegedy. *Limits of dense graph sequences.* J. Comb. Theory B 96, 2006, 933–957.

[63] L. Lovász, B. Szegedy. *Testing properties of graphs and functions.* Israel J. Math. 178, 2010, 113–156.

[64] L. Mach, T. Toufar. *Amalgam width of matroids.* 8$^{\text{th}}$International Symposium on Parameterized and Exact Computation, 2013, Lecture Notes in Computer Science 8246, Springer, 2013.

[65] A. Marcus, G. Tardos. *Excluded permutation matrices and the Stanley-Wilf conjecture.* J. Comb. Theory A 107, 2004, 153–160.

[66] J. Nešetřil, P. Ossona de Mendez. *A model theory approach to structural limits.* Preprint available as CoRR, arXiv:1303.2865, 2013.

[67] J. Nešetřil, P. Ossona de Mendez. *A unified approach to structural limits, and limits of graphs with bounded tree-depth.* Preprint available as CoRR, arXiv:1303.6471, 2013.

[68] J. Nešetřil, P. Ossona de Mendez. *Sparsity: graphs, structures, and algorithms.* Springer-Verlag, 2012.

[69] S. Oum, P. D. Seymour. *Approximating clique-width and branch-width.* J. Comb. Theory B 96, 2006, 514–528.

[70] S. Oum, P. D. Seymour. *Certifying large branch-width.* Proc. SODA, 2006, 810–813.

[71] J. Oxley. *Matroid Theory.* Oxford Graduate Texts in Mathematics 21, Oxford University Press, Oxford, 2011.

[72] H. Räcke. *Optimal hierarchical decompositions for congestion minimization in networks.* Proc. STOC, 2008, 255–264.

[73] A. Razborov. *Flag algebras.* Journal of Symbolic Logic 72, 2007, 1239–1282.

[74] A. Razborov. *On the Minimal Density of Triangles in Graphs.* Comb. Probab. and Comput. 17, 2008, 603–618.

[75] N. S. Robertson, P. D. N. Seymour. *Graph minors. X. Obstructions to tree-decomposition.* J. Comb. Theory B 52, 1992, 153–190.

[76] N. S. Robertson, P. D. N. Seymour. *Graph Minors. XX. Wagner's conjecture.* J. Comb. Theory B 92, 2004, 325–357.

[77] D. Rotem. *Stack-sortable permutations.* Discrete Math 33, 1981, 185–196.

[78] C. Schensted. *Longest increasing and decreasing subsequences.* Classic Papers in Combinatorics, 1987, 299–311.

[79] P. Seymour. *Recognizing graphic matroids.* Combinatorica 1, 1981, 75–78.

[80] R. Simion, F. W. Schmidt. *Restricted permutations.* European J. Combinatorics 6, 1985, 383–405.

[81] Y. Strozecki. *Monadic second-order model-checking on decomposable matroids.* Discrete Appl. Math. 159, 2011, 1022–1039.

[82] S. Vialette's plenary talk at Permutation Patterns 2014.

[83] H. Wilf's address to the SIAM meeting on Discrete Mathematics in 1992.